

Implementing Event Processing with CICS



Instrument your CICS applications for business monitoring



Integrate CICS with WebSphere Business Events



Learn how to extend CICS event processing for your environment



Chris Rayns
Michael Baylis
Ann Collins
Hannah Said
Phil Bareham
Steve Bolton
Lee Gavin
Phil Lee
Jackie Scott
Tommy Joergensen



International Technical Support Organization

Implementing Event Processing with CICS

September 2009

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (September 2009)

This edition applies to Version 4, Release 1, CICS Transaction Server.

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team who wrote this book	xii
Become a published author	xiv
Comments welcome	xiv
Part 1. Introduction	1
Chapter 1. Introduction to event processing	3
1.1 What is an event?	4
1.2 What is event processing?	4
1.2.1 Simple events	5
1.2.2 Complex events	5
1.3 Why you need events	6
1.4 Business application events and system events	6
1.5 IBM solutions for business event processing	7
1.5.1 CICS Transaction Server	7
1.5.2 CICS Explorer	8
1.5.3 WebSphere Business Events	8
1.5.4 WebSphere Business Monitor	9
1.5.5 WebSphere Enterprise Service Bus	10
1.5.6 WebSphere Message Broker	11
1.6 IBM solution for system events	12
1.6.1 Tivoli OMEGAMON XE for CICS on z/OS	12
1.7 Solutions reviewed	12
Chapter 2. CICS Event processing	15
2.1 Why emit events from CICS applications?	16
2.2 How CICS Event processing works	16
2.3 CICS Event Binding Editor	17
2.4 Event specification	19
2.5 Capture specification	20
2.5.1 Capture Point	21
2.5.2 Filter and predicates	23
2.5.3 Information sources	24
2.6 Event binding	27
2.7 Non-invasive events or SIGNAL EVENT	28

2.7.1 Automatic Capture Specification for SIGNAL EVENT	29
2.8 Event Processing Adapters	32
2.8.1 TS Queue EP adapter	34
2.8.2 Transaction start EP adapter	34
2.8.3 WMQ queue EP adapter	35
2.8.4 Custom (user written) EP adapter	36
2.9 Exporting event schema or copybook	36
2.10 EP Adapter advanced options	37
2.10.1 Dispatch priority	37
2.10.2 Transaction ID	38
2.10.3 User ID	38
2.10.4 System ID	38
2.10.5 Transactional events	38
2.11 Bundles	39
2.12 Deploy a bundle to zFS	40
Part 2. Implementing CICS event processing	43
Chapter 3. Environment overview	45
3.1 Example environment	46
3.2 Shopping sample application	47
3.3 Sample scenarios	49
3.3.1 Scenario: Query versus sale	49
3.3.2 Scenario: Stock low	50
3.3.3 Scenario: Shipped order meets service level agreements	51
3.3.4 Scenario: High-value order breakdown	52
3.4 The events emitted in the scenarios	52
Chapter 4. Setting up CICS for events	57
4.1 Our software versions	58
4.2 CICS development setup	58
4.2.1 Obtaining the CICS Explorer	59
4.2.2 CICS Explorer connectivity	59
4.2.3 Obtaining the CICS event binding editor	61
4.3 CICS System setup	61
4.3.1 Adding WebSphere MQ support to CICS region, to use the WMQ EP adapter	61
4.3.2 Enabling CICS event processing	63
4.3.3 Stopping CICS event processing	64
4.4 Creating and installing a bundle definition	66
4.4.1 Creating a new bundle definition (BUNDEF resource)	67
4.4.2 Installing a bundle definition into CICS	71
4.5 Enabling and disabling and discarding events	74
4.5.1 Discarding a bundle	80

4.5.2 Replacing a deployed bundle	82
4.6 Security considerations for CICS events	82
4.6.1 Changes to security	82
4.6.2 Setting up CICS security for event bindings	83
4.6.3 The user ID in event bindings	87
Chapter 5. Generating events	91
5.1 Creating the HFS directories	92
5.2 Creating the bundle project	92
5.3 Defining the Temporary Storage queue adapter	107
5.4 Exporting the event specification	108
5.5 Exporting the bundle project	110
5.6 Installing the bundle in CICS	114
5.7 Testing the Shopping application	119
5.8 Verifying the event processing	120
5.9 Creating events to the Transaction Start EP adapter	122
5.10 Creating event to WebSphere Business Events	127
5.11 Adding event specifications	130
5.11.1 Fulfill event specification	135
5.11.2 The Ship event specification	139
5.11.3 The SendOrder event specification	140
5.12 Creating events for WebSphere Business Monitor	142
Chapter 6. Troubleshooting	149
6.1 Best practices for performance	150
6.1.1 Capturing an event	150
6.1.2 Dispatching the EP adapter	151
6.1.3 Emitting an event through the EP adapter	151
6.2 Monitoring and statistics	152
6.2.1 Monitoring	152
6.2.2 Statistics	153
6.3 Problem determination	157
6.3.1 Events not captured	158
6.3.2 Unexpected events captured	165
6.3.3 Capture data is not as expected	167
6.3.4 Captured events are missing from WebSphere Business Events ..	168
6.3.5 Captured events are missing from WebSphere Business Monitor ..	170
Part 3. Scenarios	181
Chapter 7. WebSphere Business Events scenario	183
7.1 Development setup, WebSphere Business Events tooling, and so forth	184
7.1.1 Scenario description: Interaction between multiple CICS events ..	184

7.1.2 CICS parts: Event specs, export event schemas (WBE)	188
7.1.3 WebSphere Business Events development tooling	191
7.1.4 WMQ setup and WebSphere Business Events configuration for WMQ.	193
7.1.5 WebSphere Business Events Configuration for actions	199
7.1.6 Running the WebSphere Business Events scenario.	202
Chapter 8. WebSphere Business Monitor scenario	213
8.1 Scenario description	214
8.2 CICS parts: Event specs, export event schemas	214
8.3 WMQ set up, WMQ on z/OS through SIB to CEI	214
8.3.1 Installing the mediation EAR file	215
8.3.2 Running the configuration script	219
8.3.3 WebSphere MQ queue manager configuration.	227
8.3.4 Verify the configuration	232
8.3.5 Security considerations	234
8.4 Create monitor model	235
8.4.1 Import event definition schemas	237
8.4.2 Creating the monitor details model	241
8.4.3 Create KPI and Dimensional Models	269
8.5 Generate and deploy monitor model	279
8.6 Create Business Space dashboards	283
8.7 Test and verify results	286
8.8 Troubleshooting WebSphere Business Monitor	296
8.8.1 Events arrived from CICS but not delivered to monitor model	296
8.8.2 Events delivered to monitor model but not showing up in the dashboards	303
8.8.3 Metric values not being set or being incorrectly set	304
Chapter 9. Custom EP adapters	307
9.1 Why write a custom EP adapter?	308
9.2 How to write a custom EP adapter	308
9.2.1 EP adapter samples	308
9.2.2 The CICS event object	310
9.2.3 Event formatting	319
9.2.4 Event emission	321
9.3 Running the sample EP adapter	322
9.3.1 EP adapter program EPCUSTOM	322
9.3.2 Event binding ShoppingCustomBinding	322
9.3.3 CICS resource definitions	327
9.3.4 Generating custom events	337
Chapter 10. Updating the monitoring to use custom EP adapter events.	339

10.1 Create event definitions and event parts	340
10.2 Update keys and metrics	344
10.3 Deploy and test monitor model	344
Appendix A. Additional material	347
Locating the Web material	347
Using the Web material	348
System requirements for downloading the Web material	348
How to use the Web material	348
Related publications	349
IBM Redbooks	349
Other publications	349
Online resources	349
How to get Redbooks	350
Help from IBM	350

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AlphaBlox®
BetaWorks™
CICSplex®
CICS®
DataPower®
DB2®
FileNet®

IBM®
iSeries®
MVS™
OMEGAMON®
OS/390®
RACF®
Rational®

Redbooks®
Redbooks (logo) ®
System z™
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

FileNet, and the FileNet logo are registered trademarks of FileNet Corporation in the United States, other countries or both.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Interchange, and the Shadowman logo are trademarks or registered trademarks of Red Hat, Inc. in the U.S. and other countries.

SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

J2EE, Java, JavaScript, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Expression, Microsoft, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Governance concerns the methods that businesses use to control important aspects of their business. These methods must be measured. Compliance is the tolerance to the criteria that have been set. CICS v4 allows you to configure your system to produce application compliance data feeds and events without application change to show solution compliance on a dashboard.

The governance theme will largely be linked to CICS TS v4.1, which allows you to generate business events without requiring application change. This new capability will allow you to populate business dashboards provided by WebSphere® Business Monitor and to search for patterns in events with WebSphere Business Events. For business event processing, when CICS TS is operating as a stand-alone system, or used in conjunction with WebSphere Business Monitor or WebSphere Business Events (or both), it will enable you to understand and manage your business more easily, and to monitor risk and business compliance more effectively.

This IBM® Redbooks® publication, is split into 3 parts:

Part 1 introduces event processing. We explain what it is and why you need it. We also review the CICS TS implementation of event processing.

Part 2 of the book focuses on implementation of event processing in CICS. It gives a step-by-step guide to implementing CICS event processing, along with the environment we used. We also provide some ideas on troubleshooting.

Part 3 runs through our scenarios.

- ▶ Scenario 1: The CICS COBOL application emits the events QueryStock, Order, fulfill, and Ship. These four events are defined in a single CICS event binding file and are emitted through a single event adapter. For these events to be consumed by WBE, they are required to be sent from CICS to WBE through WMQ
- ▶ Scenario 2: The main focus of our business activity monitoring (BAM) is the customer. We will be monitoring the stock querying and buying behavior of our customers. We will also be tracking key performance indicators (KPIs), which are the metrics that used to quantify and measure business performance against strategic and operational business targets (such as tracking actual order fulfillment durations against targets).

- Scenario 3: We discuss why you might need a custom event processing adapter, describe how to write a custom event processing adapter and provide a sample custom event processing adapter that emits simplified CICS events to WebSphere Business Monitor over HTTP.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Hursley Center.

Chris Rayns is an IT Specialist and the CICS project leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively on all areas of CICS. Before joining the ITSO, Chris worked in IBM Global Services in the United Kingdom as a CICS IT Specialist.

Michael Baylis is a tester in the CICS Transaction Server development organization based at the Hursley laboratory in the UK.

Ann Collins is a software engineer in the CICS Transaction Server development organization based at the Hursley laboratory in the UK. She has over twenty years software development experience of both z/OS® and workstation products. Most recently, Ann was the development team lead for the CICS TS 4.1 event processing runtime support. Prior to joining IBM, Ann worked as a CICS application and systems programmer.

Hannah Said is a Senior WebSphere IT Specialist in the IBM BetaWorks™ organization, and is based at the Hursley laboratory in the UK. Hannah specializes in WebSphere Business Monitor, WebSphere Business Events, and CICS. For the last three years, she has been working with customers and ISVs during beta programs, supporting customer engagements and delivering enablement to the IBM field organizations. Prior to this she had four years of experience in CICS. Her roles included CICS TS Level 3 Service, CICS Tools Project Manager, and technical focal point for the CICS TS 3.1 Beta Program.

Phil Bareham is a Senior IT Specialist working for the Worldwide Technology Practice, which is part of IBM Software Services for WebSphere (ISSW), based out of the Hursley laboratory in the UK. For the last seven years, he has been working with customers helping them to configure and implement WebSphere MQ, Message Broker, Workflow, Application Server, and Process Server. Most recently, Phil has added WebSphere Business Events to his portfolio of products. Phil has covered all of these products on z/OS and non-z/OS platforms. Prior to joining ISSW, Phil worked as a CICS application programmer both inside and outside IBM.

Steve Bolton is a Software Engineer working for the CICS Explorer Team on the Event Binding Editor. He is based in the Hursley laboratory in the UK. Steve has worked with CICS for many years as an application programmer, system programmer, software engineer, and tester. Steve has also developed software for z/OS, Windows®, Linux®, and iSeries®.

Lee Gavin is a Consulting IT Specialist in the WebSphere BPM Black Belt Team. The Black Belt Team is a worldwide team with a focus on customer pre-sales engagement and enablement for the IBM WebSphere BPM portfolio of products. Prior to joining the Black Belt Team, Lee was the SW IOT TechWorks lead for WebSphere Business Monitor, WebSphere Message Broker, and WebSphere Adapters. From 2001–2006, Lee was an ITSO Project Leader (in Hursley and Raleigh) in the WebSphere team, specializing in business integration and application integration.

Phil Lee is a Software Support L2 Specialist working for IBM United Kingdom, based in Farnborough. He has worked in IBM since 1984, providing technical support for CICS products, including CICS TS, CICSplex/SM, and CICS® tools. During the last few years he has completed a number of assignments with the ICS L3 Change Team in Hursley. He holds a degree in Mechanical Engineering from the Lanchester Polytechnic, Coventry and is a Chartered IT Professional.

Jackie Scott is a Software Test Engineer at the Hursley laboratory, England and has 24 years of experience in the IT industry, starting out as a programmer in a mainframe environment. Jackie joined IBM as a Software Support Specialist providing customer support for MVS™ and OS/390®, followed by several years as an MVS systems programmer. After four years in the Java™ Technology Centre at Hursley, Jackie joined CICS Development, also at Hursley, as a Software Tester.

Tommy Joergensen is a Senior IT Specialist working for IBM System Technology Group in IBM Denmark. He has more than 30 years of experience working in different areas of CICS technical support, including 3 years at IBM Hursley. For the last two years he has worked in the System z™ New Technology Center and System z Benchmark Center in Montpellier supporting CICS TS and WebSphere Application Server for z/OS.

Thanks to the following people for their contributions to this project:

Robert Haimowitz
International Technical Support Organization, Hursley Center

Chris Backhouse, ILOG Synergies and Integration Team
IBM Hursley

Catherine Moxey, CICS TS Senior Developer
IBM Hursley

Daniel Millwood, Software Developer
IBM Hursley

Mark Hiscock, WebSphere Business Events Development
IBM Hursley

Peter Crocker, Development Lead, Architect, Business Event Processing
IBM Hursley

Francis Burgess, Software Product Introduction Specialist
IBM Hursley

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an e-mail to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400



Part 1

Introduction

In this part of this IBM Redbooks publication, we introduce event processing, explain what it is, and why you need it. We also review the CICS implementation of event processing.



Introduction to event processing

In this chapter, we define what we mean by event processing and explain the distinctions between simple and complex events.

We show why event processing is useful to your business, describe the difference between business and system events, and conclude with a summary of IBM products for event processing.

1.1 What is an event?

An event is something that happens that is significant to a system. Here are some examples:

- ▶ Open a bank account.
- ▶ Sense a temperature change.
- ▶ Click a mouse button.
- ▶ Browse an inventory without making a purchase.
- ▶ An unusual history of purchases on a credit card.

For this book *event* is the term used to describe an electronic message indicating a change in the state of an enterprise. An event has a name and usually some data, sometimes referred to as the event payload.

Events are generated and processed asynchronously in near real-time. The processing of an event is de-coupled from the computer operations that cause it to be emitted.

1.2 What is event processing?

Event processing is the capture, enrichment, formatting and emission of events, the subsequent routing and any further processing of emitted events (sometimes in combination with other events), and the consumption of the processed events.

Events can be produced throughout a business enterprise. At the edges of the enterprise, events can be detected by sensors. In the enterprise network, events can be produced when business processes start and then either complete or fail. The activity of the enterprise and its business can be monitored and changed as a result of events. Event processing consists of three main steps:

- ▶ Event sources emit events into the event processing system. Examples of event sources are simple Radio Frequency Identification (RFID) sensors and actuators, business flows, and CICS applications. The event processing system can perform a variety of actions on events:
 - Simple enriching of the event (for example, adding a time stamp to the event data).
 - Adding information about the source of the event.
 - Processing multiple simple events, from multiple event sources, against event patterns to produce a new derived event. Processing of this kind is often referred to as complex event processing.

CICS event processing provides filtering, capture, enrichment, formatting, and routing of single business events, enabling CICS to act as a source of single business events.

- ▶ The event resulting from processing is made available for consumption.
- ▶ The event consumer reacts to the event. The event consumer might be simple and just update a database or a visual dashboard with the data carried with the event, or it might carry out new business processing based on the event.

We categorize these events as simple or complex.

1.2.1 Simple events

The first three examples in the list in 1.1, “What is an event?” on page 4 are what we call simple events. For many years, organizations have been using simple event processing to detect and respond to discrete events (for example, when customers open a new account sending them a welcome letter explaining other facilities provided by the bank).

1.2.2 Complex events

The last two examples in the list in 1.1, “What is an event?” on page 4 describe what we call complex or derived events that are obtained by looking at the patterns of simple events over time. To detect customers browsing an inventory without making a purchase we could emit events when they perform the following actions:

- ▶ Browse an item in the inventory.
- ▶ Buy an item.

Detecting the patterns from these simple events can produce the complex event in example four in 1.1, “What is an event?” on page 4.

In large organizations, tens of millions of events occur every day, but not all events are of equal importance. Greater insight can be obtained when a pattern of related or seemingly unrelated events from one or more sources is detected, and responses to that pattern are coordinated. Considerable complexity, time, and cost would be involved in writing custom code for such a solution. This can be replaced by general software technology that is designed to detect event patterns and coordinate responses, sometimes known as complex event processing software.

1.3 Why you need events

Events allow businesses to be more responsive and flexible, and to address governance and compliance concerns.

The mainstream adoption of service-oriented architecture (SOA) has opened new opportunities for highly responsive business solutions. SOA brings greater flexibility to business processes, and helps bring business and IT in line with each other. Enterprises are challenged by seeking to maximize SOA solution advantages (such as speed to market), while complying with business controls, industry standards, and government legislation.

Business governance and compliance are increasingly important in many industries. These terms cover a crucial range of issues including financial transparency, information privacy and process control. The Sarbanes-Oxley Act, the Health Insurance Portability and Accountability Act (HIPAA), or Basel II and their associated information requirements are just some of the standards required for business compliance and governance.

Governance describes a formalization of the decision-making processes within an organization. It can cover many aspects of business and depends on accurately maintaining and auditing which decisions can be freely made, which ones need specific approvals and determines who can make them.

Compliance is about ensuring adherence to mandated standards and governance policy. This includes the definition of information about which governance decisions are based, and maintaining accurate operational control to ensure that business application execution meets the required enterprise, industry, and government standards.

1.4 Business application events and system events

CICS event processing is designed for business application events. All events in an enterprise can be seen as having a business consequence and so can be described as business events. Whether the event and event processing are specified by a line-of-business manager or an IT programmer, the event relates to the business application or system that is running. It therefore has business relevance. For example, an event that implies imminent failure of a system running an order-processing application could be considered relevant to the line-of-business manager. In general, the procedures that emit and consume business events are distinct from those that process IT infrastructure-related events.

System-level events generally have a technical focus and relate to monitoring operating system, application execution, and middleware running on the system. The event data is usually equally technical, specifying the identifiers of the resources under observation.

Business application events are usually related to higher-level business processes. They specify conditions in terms of what the application is doing for the business. For example, contrast the business event “a new order has been placed” with the system events that are used to assess the compute time for processing the order: inventory file opened, order information storage released, and so forth.

The consumers of business events are often required to be independent of the implementation specifics of the systems that emit the events. For example, it is possible for one event consumer to process events from several disparate ordering systems and provide a single consolidated view of the business application’s state.

In general, we consider system events and business events as distinct event types with different software solutions and audiences.

1.5 IBM solutions for business event processing

The IBM software portfolio enables a range of options for integrated processing of business events.

1.5.1 CICS Transaction Server

IBM CICS Transaction Server (TS) business applications are the main source of business information in most large enterprises. The CICS runtime detects instances of events that are enabled, and captures the events and payload without the need to make application code changes. CICS Event Processing is a core component of the CICS runtime, and provides all the qualities of service you would expect of CICS. When CICS captures events, it carries out specified filtering, enriches the event with information about the application context in which it occurred, formats the event and routes it to the appropriate event consumer.

It is possible to emit events in formats suitable for consumption by WebSphere Business Events, WebSphere Business Monitor, and other consumers.

CICS Event Processing support is extensible, with options for customization.

See *CICS Transaction Server for z/OS, Version 4 Release 1* at the following Web page:

<http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp>

1.5.2 CICS Explorer

IBM CICS Explorer is the new face of CICS, and an integration point for CICS tooling with rich CICS views, data, and methods. It provides a common, intuitive, and Eclipse-based environment for architects, developers, administrators, system programmers, and operators.

CICS Explorer has the following features:

- ▶ Provides task-oriented views giving integrated access to a broad range of data and control capabilities
- ▶ Has powerful, context-sensitive resource editors
- ▶ Is an integration point for CICS TS, CICS Tools, CICS Transaction Gateway, Problem Determination Tools, and Rational® Tools

For CICS event processing, the CICS Explorer contains the CICS event binding editor, which is used to perform the following tasks:

- ▶ Define events to be emitted and their payload data.
- ▶ Specify to the CICS runtime how to detect when the events occur.
- ▶ Indicate how events are to be formatted and routed.
- ▶ Deploy the event definitions to zFS for installation into CICS.

See *CICS Explorer* at the following Web page:

<http://www-01.ibm.com/software/http/cics/explorer/>

1.5.3 WebSphere Business Events

IBM WebSphere Business Events (WBE) is an IBM software product designed to support business event processing by meeting the high-volume demands and processing required across industries and application domains. WBE provides graphical, codeless user interfaces that simplify implementation and empower business users to develop and maintain event processing logic.

WBE consists of the following basic constructs:

- ▶ Connectivity to business events
- ▶ Event processing engine for evaluating and detecting event patterns
- ▶ Initiation of business responses (actions)

Business events can exist anywhere within the extended computing infrastructure, both inside and outside the firewall. Events can be communicated directly between systems or pushed into the communications backbone for use by any system. A message-based publish-or-subscribe or request-or-reply transport such as IBM WebSphere MQ is an ideal transport infrastructure for event processing.

Based on user definitions, the WebSphere Business Events processing engine detects and sifts through the mass of events occurring across the information infrastructure, identifying only those events and patterns of interest. Upon detecting a defined event or pattern (actionable situation), the engine initiates one or more business responses (actions).

Responses range from sending electronic alerts to initiating the execution of follow-on processes. These actions are communicated directly to systems (or over the communications backbone), indicating that an actionable event or pattern has been detected.

See *WebSphere Business Events* at the following Web page:

<http://publib.boulder.ibm.com/infocenter/wbevents/v6r2m1/index.jsp>

1.5.4 WebSphere Business Monitor

IBM WebSphere Business Monitor (WBM) is a comprehensive business activity monitoring (BAM) product that provides business users and managers with a real-time and end-to-end view of business processes, events, and operations. WBM aggregates and correlates events into metrics that give objective measurements on the status of business processes.

WBM shows business users real-time information about the performance of critical business processes. It offers user-friendly and customizable dashboards that enable complete insight into the business flowing through the system. These dashboards can calculate and display key performance indicators (KPIs) and metrics derived from the following sources:

- ▶ Business processes
- ▶ Business activity data
- ▶ Business events

Business users can view these KPIs, metrics, events, and alerts through various means, including lightweight Web interfaces, Smartphones, corporate portals, and on desktops. These options give business users immediate actionable information and insight into their business operations to mitigate risk and take advantage of opportunities.

The following list details some of the real-time events from which WBM enables collection:

- ▶ CICS
- ▶ IBM Business Process Management (BPM) Suite and Connectivity portfolio:
 - WebSphere Process Server
 - IBM FileNet® P8 Business Process Manager
 - WebSphere MQ Workflow
 - WebSphere Business Events
 - WebSphere Message Broker
 - WebSphere Enterprise Service Bus
 - WebSphere DataPower® Integration Appliance XI50
 - WebSphere Partner Gateway
- ▶ Events and data through the use of WebSphere Adapters and the IBM Connectivity portfolio from the following sources:
 - Oracle®
 - SAP®
 - Siebel®
 - Other enterprise resource planning (ERP) and CRM applications
- ▶ Other third-party applications through the IBM Connectivity portfolio

A bi-directional flow of events between WBM and WBE is enabled. Customers can use a single dashboard to view performance of business processes through KPIs, and view any alerts generated by WBE to get real-time insight into what is happening within the organization. Similarly, the bidirectional event flow enables customers to feed any alerts on processes or business generated by WBM to WBE for detecting patterns within those alerts which may otherwise go undetected; and to initiate follow on processing.

See *WebSphere Business Monitor* at the following Web page:

<http://www-01.ibm.com/software/integration/wbimonitor/>

1.5.5 WebSphere Enterprise Service Bus

IBM WebSphere Enterprise Service Bus (WESB) provides Web services connectivity and Java Message Service (JMS) messaging, improving flexibility through the adoption of service-oriented interfaces.

WESB provides a smart approach to SOA, delivering a standards-based connectivity and integration solution that allows you to create and deploy interactions quickly and easily between applications and services, with a reduced number and complexity of interfaces.

WebSphere Enterprise Service Bus provides the following features:

- ▶ Offers easy-to-use tools that require minimal programming skills and is simple to install, configure, build, and manage.
- ▶ Supports hundreds of independent software vendor solutions through WebSphere adapters.
- ▶ Reconfigures dynamically to meet changing business processing loads.
- ▶ Provides easy interactions with any JMS and HTTP applications.

See *WebSphere Enterprise Service Bus* at the following Web page:

<http://www-01.ibm.com/software/integration/wsesb/>

1.5.6 WebSphere Message Broker

IBM WebSphere Message Broker is built for universal connectivity and transformation in heterogeneous IT environments. It distributes information and data generated by business events in real time to people, applications, and devices throughout your extended enterprise and beyond.

WebSphere Message Broker offers the following features:

- ▶ Provides a smart approach to SOA, extending the reach of your business beyond your firewall by supporting a broad range of multiple transport protocols and data formats
- ▶ Integrates multiple applications, networks, and device types using a platform-independent-based enterprise service bus (ESB) that lets you conduct business reliably and securely
- ▶ Increases business agility and flexibility, extending easily to a federated ESB model, while reducing development costs by separating integration logic from applications
- ▶ Improves the flow of information around the business, moving away from hard-coded point-to-point links to more flexible distribution mechanisms such as publish/subscribe and multi-cast.
- ▶ Uses a simple programming model for connectivity and mediation, including a robust set of pre-built mediation function and ways to customize mediations
- ▶ Exploits the industry-leading WebSphere MQ messaging infrastructure, and supports transformation options with graphical mapping, Java, ESQL, XSL, and WebSphere Transformation Extender
- ▶ Delivers extensive administration and systems management facilities for developed solutions

See *WebSphere Message Broker* at the following Web page:

<http://www-01.ibm.com/software/integration/wbimessagebroker/>

1.6 IBM solution for system events

In this section we discuss the IBM solution for systems events, Tivoli® OMEGAMON® XE for CICS on z/OS.

1.6.1 Tivoli OMEGAMON XE for CICS on z/OS

Although this book is about business application event processing, we mention an available solution for processing system events for CICS.

IBM Tivoli OMEGAMON XE for CICS on z/OS enables monitoring and management of CICS transactions and resources. It quickly detects and isolates problems when they occur on your complex CICS systems to minimize or eliminate any impact to your customers and your business.

See *IBM Tivoli OMEGAMON XE for CICS Transaction Gateway on z/OS: User's Guide*, SC23-5963.

1.7 Solutions reviewed

With our review of some of the products involved in event processing, we consider the question of which products you should use in which situations.

If your business processing is running within CICS, that will be the source of your events, and forms the subject of this IBM Redbooks publication. There will be situations in which the actions to be taken as a result of the events also involve processing within CICS. In other situations, you will want to involve other products.

If you want to monitor the processing which is happening within CICS, to look at key performance indicators, to provide a dashboard to allow business users to understand the behavior of the business, and to receive alerts, you could use WebSphere Business Monitor.

If you want to derive additional information from combinations of events, potentially including events from other sources in addition to CICS, or to consider events over time, you can use WebSphere Business Events.

You might also want to monitor processing based on some of these derived events, in which case the WebSphere Business Events can look for the patterns of interest, and can send the resulting events to WebSphere Business Monitor.



CICS Event processing

In this chapter, we explain why CICS Event processing (EP) is useful. We discuss the implementation of event processing in CICS, which uses event specifications, capture specifications, and adapter information contained in an event binding.

This chapter introduces the event binding editor by including snapshots of the CICS Catalog sample as we explain the concepts in CICS event processing. To explore the catalog sample, download the CICS Explorer and use the wizard to generate the sample binding automatically.

In addition, we show how to deploy event bindings to zFS using a CICS Bundle.

2.1 Why emit events from CICS applications?

Given the considerable amount of business processing that is carried out in CICS systems across the world (over 30 billion transactions a day), CICS is a significant source of business events.

This can provide enhanced business flexibility and help to meet governance and compliance regulations, as described in 1.3, “Why you need events” on page 6.

2.2 How CICS Event processing works

Using event specifications defined to CICS, events can be captured from existing business application programs without altering the original code. Figure 2-1 shows an overview of the process.

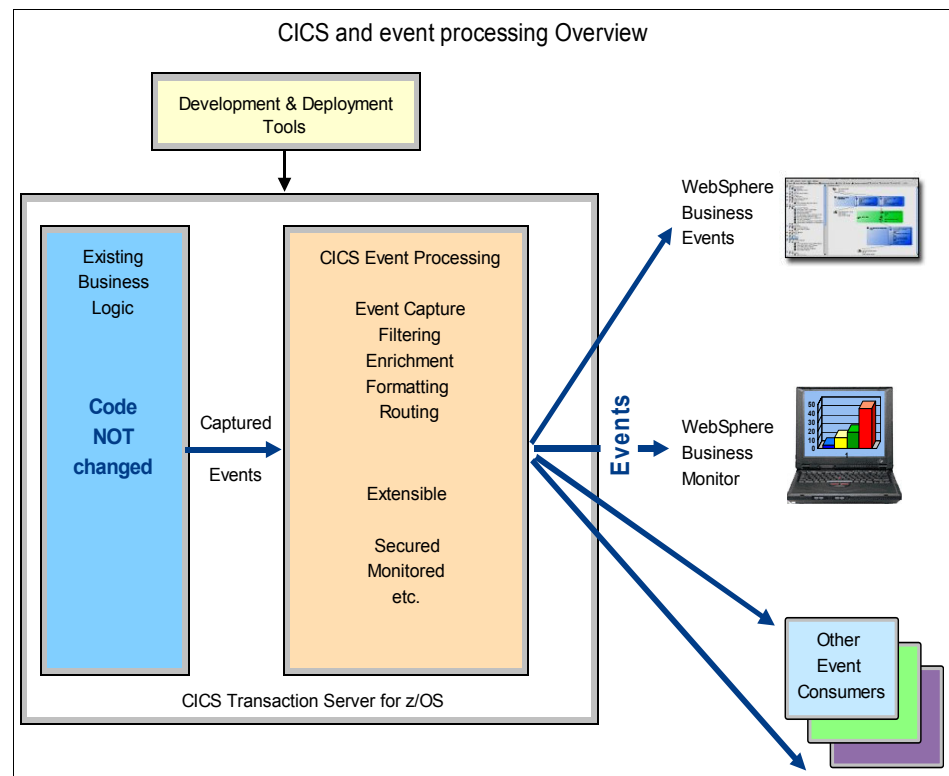


Figure 2-1 CICS and event processing overview

During execution of the business application, many potential capture points occur. These capture points include relevant CICS Application Programming Interface (API) calls and when a program starts. Each time a program executes a capture point, CICS checks each of the enabled capture specifications that match the capture point.

Each matching capture specification contains optional filters to be compared against the application context, some command options on the API call, and data passed on the API call.

If the filters match, CICS collects the payload information from information sources described in the capture specification, enriches it with context data, then queues the event for dispatch so that the application can continue to execute quickly.

A separate process in CICS routes the event and any payload data through the event processing adapter described in the event binding. Events with a high dispatch priority are routed first. Events marked as transactional are only emitted after the transaction reaches a sync point and commits. The EP adapter then emits the event to a consumer such as WebSphere Business Events or WebSphere Business Monitor.

2.3 CICS Event Binding Editor

Use the CICS Event Binding Editor in the CICS Explorer resource view to create an event binding that contains one or more event specifications (See 2.4, “Event specification” on page 19).

Deploy a bundle which contains the event specification to zFS and install it in CICS. (See 2.11, “Bundles” on page 39).

We show panel snapshots from the editor for the catalog sample when discussing the concepts here, because the editor can automatically generate the catalog sample for you. You may download the CICS Explorer onto your workstation through the CICS Explorer website:

<http://www-01.ibm.com/software/http/cics/explorer/>

To generate the catalog sample in the editor, perform the following steps:

1. Switch to Explorer Resource perspective. If you do not see the buttons for Resource and CICS SM shown, click the plus (+) button on the left or use **Window Open Perspective Other** to show it. See Figure 2-2.



Figure 2-2 Explorer Resource Perspective

2. Create a CICS Bundle project. See Figure 2-3.

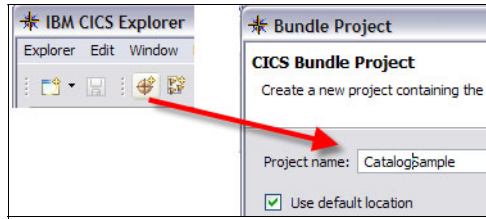


Figure 2-3 Creating a CICS Bundle project

3. Generate the catalog sample event binding file. See Figure 2-4.

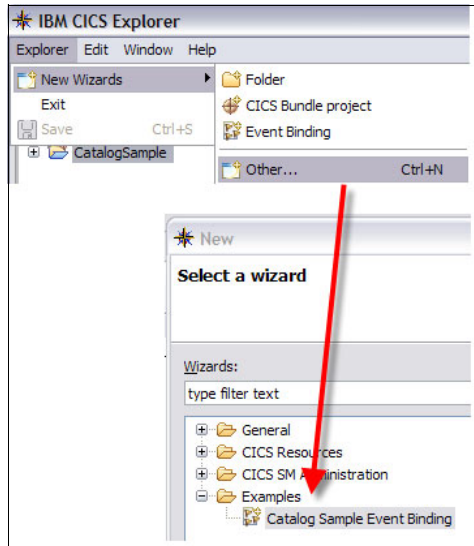


Figure 2-4 Catalog sample binding file

2.4 Event specification

An event specification defines a business event to CICS. An event specification can be created using the CICS event binding editor by business analysts and developers or by an application analyst in response to a business requirement. An event specification describes an event and its processing in natural language.

See Figure 2-5, which shows the components of an event specification.

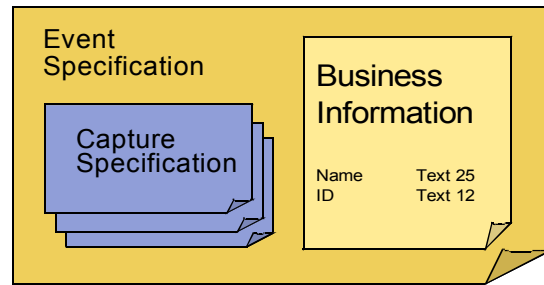


Figure 2-5 Event Specification components.

An event specification also defines the business information to be contained in the event (if a payload is required). Example 2-1 is an example of a event specification.

Example 2-1 Sample event specification

When The stock level is low and there is no re-order in place, capture:

- Progam name (text)
- Item Ref (numeric)
- Item Description (text)
- In Stock (numeric)
- On Order(numeric)

The event specifies the order in which the payload is produced.

The event description can also indicate the intended use for this event, such as sending the data to the business orders dashboard.

The event specification contains one or more capture specifications. See 2.5, "Capture specification" on page 20 for more information about capture specifications.

Note: The event specification represents what the event is. The capture specification represents when and how to capture it.

Figure 2-6 shows the event specification for the catalog sample.

Specifications

General
Identify and describe the event.

Name: Catalog_stock_status_check

Description: The stock level is low and there is no re-order in place.

Emitted Business Information
Describe and order the business information to be emitted by the event.

Name	Type	Length	Precision	Description
Program_name	Text	8	0	Program Name
Item_ref	Numeric	4	0	Item reference number
Item_description	Text	40	0	Item description in catalog
in_stock	Numeric	4	0	Current number of items in stock
on_order	Numeric	3	0	Number of items on order

Figure 2-6 Catalog sample event specification

You need at least one capture specification to detail where to capture the event. However, there may be more than one place in your application where the event can occur and it is represented by different capture points. For example, in a stock application, orders may be placed through an online form or through a queue. These would require different capture points and the format of the incoming data might be different. In this case you can define more capture specifications to describe these additional places.

2.5 Capture specification

An application analyst with knowledge of your business applications takes a defined business event and defines one or more capture specifications to satisfy the event.

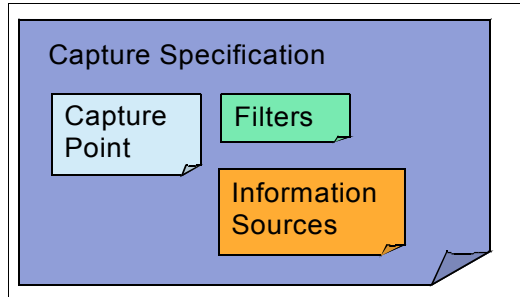


Figure 2-7 Components of a capture specification

A capture specification consists of the capture point, such as an **EXEC CICS** command, relating to the event, and some filter predicates that give more information about the exact location or locations where the event occurs. The location of the event in the application logic depends on how it is specified. If a CICS application contains two instances of the same **EXEC CICS** API command, and the filter specification does not distinguish between the two commands, an event is emitted when both instances of the command are executed.

The capture specification must contain an information source for each item of business information in the event specification.

You add a capture specification by pressing **Add a Capture Specification** on the display for an event specification, or right-click the event specification in the tree at the left and select **Add a Capture Specification** in the pop-up menu.

2.5.1 Capture Point

The capture point is the place in a CICS application where a particular event can be captured (for example, an **EXEC CICS READ FILE** command or a program starting).

The Program Initiation capture point is the only one that is not an **EXEC CICS** API call.

Table 2-1 on page 22 lists the **EXEC CICS** capture points.

Table 2-1 Capture points

CONVERSE	DELETE FILE	DELETEDQ TD
DELETEDQ TS	INVOKE SERVICE	LINK PROGRAM
PUT CONTAINER	READ	READNEXT
READPREV	READQ TD	READQ TS
RECEIVE	RECEIVE MAP	RETRIEVE
RETURN	REWRITE	SEND
SEND MAP	SEND TEXT	SIGNAL EVENT
START	WEB READ	WEB READNEXT
WRITE FILE	WRITEQ TD	WRITEQ TS
XCTL		

With most capture points if the filter is TRUE, the event is generated after the capture point has executed. For CONVERSE, INVOKE SERVICE, and LINK PROGRAM capture points, choose whether to generate the event before or after the call. For Program Initiation, RETURN and XCTL capture points, the event is always generated before the call.

Figure 2-8 shows the capture point for the catalog sample

The screenshot shows the 'Capture Point' configuration window. It has two tabs: 'Filtering' and 'Information Sources'. The 'General' section is expanded, showing the 'Name' field with the value 'Check_stock_status_on_rewrite' and an 'Edit...' button. The 'Description' field contains the text 'The number in stock and number on order are available in the FROM parameter of the REWRITE command'. Below the description is a 'Remove Capture Specification...' button. The 'Capture Point' section is also expanded, showing a dropdown menu with 'REWRITE' selected. Below the dropdown are two radio buttons: 'Capture before' (unselected) and 'Capture after' (selected). At the bottom right, there is a link that says 'Next: Filtering ->'.

Figure 2-8 Catalog sample capture point

2.5.2 Filter and predicates

The filter is a set of predicates connected by AND, used to determine whether an event is captured. If all predicates evaluate to TRUE, the event is captured. Predicates that evaluate to FALSE filter out events.

A predicate is an expression used as part of a filter, consisting of a data item, an operator, and a value. A predicate is used with data values on the API call or context data, to restrict the occasions when an event is emitted to the occurrences of interest.

In the following example of a predicate, Current Program Starts With EXAM, the data item is Current Program, the operator is Starts With, and the value is EXAM. Any program name starting with EXAM will be TRUE, like the following example:

EXAMPLE EXAM01

Depending on the capture point, you may be able to specify predicates for the application context, options on the API command, and application data.

You use the following predicates to filter by the application context:

Transaction ID	Current Program	User ID
Response Code	EIBAID	EIBCPOSN

You may use predicates to filter by application command options. For example on a SEND MAP the available command options are as follows:

MAP*	MAPSET	ALARM
------	--------	-------

At program initiation the command options are as follows:

PROGRAM*	CHANNEL
----------	---------

In these examples, MAP and PROGRAM are marked with an asterisk in the editor to indicate that they are *primary predicates*. Specify filter operators and values for primary predicates to maintain CICS performance. Primary predicates are defined for all commands with the following exceptions:

CONVERSE, RECEIVE, RETRIEVE, RETURN, SEND TEXT, WEB READ, and WEB READNEXT

When using capture points for API calls that pass data between the application and CICS, you may specify predicates for the application data. You can import a language structure for the application data to help specify the type of data, the offset into the application data and the length and precision if applicable. For example, on an XCTL capture point, the application data areas are COMMAREA and CHANNEL.

The command will either use a COMMAREA or CHANNEL, so you cannot define application data for both data areas in one capture specification.

Figure 2-9 shows filtering for the catalog sample

Filtering | Capture Point | Information Sources

Application Context
Define predicates to filter events.

Context	Operator	Value
Transaction ID	Equals	EGUI
Current Program	Equals	DFH0XVDS
User ID	All	
Response Code	Equals	Ok

Application Command Options
Define predicates for command options. Predicates marked with * should be specified to maintain CICS performance.

Name	Operator	Value
FILE*	Equals	EXMPCAT

Application Data
Define predicates for application data. Whilst defining a predicate you may import a language structure and choose an item from it.

Source	Container	Offset	Length	Operator	Value
FROM		53	4	Less Than	0024
FROM		57	3	Equals	000

[<- Back: Capture Point](#) [Next: Information Sources ->](#)

Figure 2-9 Catalog sample filtering

2.5.3 Information sources

If you want to supply a payload for the event, perform the following tasks:

- ▶ Add business information items to the event specification in the order you want them emitted.
- ▶ Define an information source for each item of business information in the capture specifications for the event.

Figure 2-10 on page 25 shows information sources for the catalog sample.

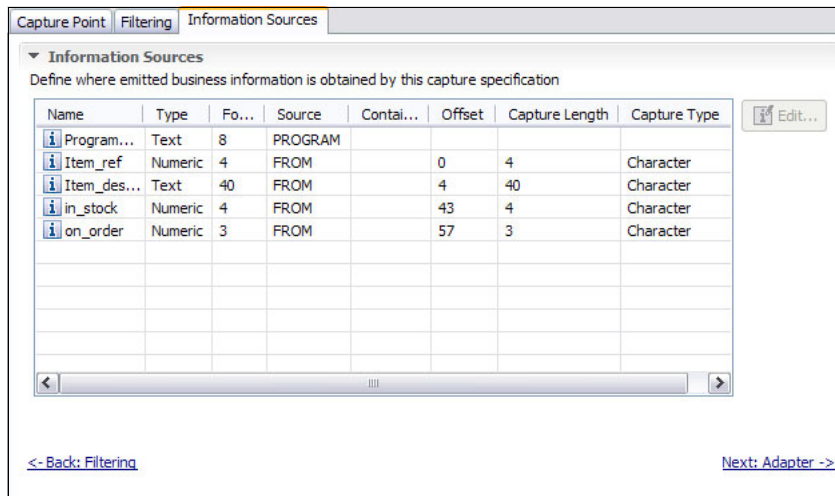


Figure 2-10 Catalog sample information sources

You add an information source by choosing from the following options:

- ▶ Application context
- ▶ Application command options
- ▶ Application data

Application context

Choose from USERID, PROGRAM, or TRANSID. CICS knows the format and length of these choices.

Application command options

These are the values of options passed on the API command. For example at the INVOKE SERVICE capture point the choices for the application data areas are as follows:

SERVICE	OPERATION	URI
CHANNEL	URIMAP	

CICS knows the format and length of these choices. So if you chose SERVICE for an application data area, CICS would supply a 32 character value, which is the length of the SERVICE parameter on the INVOKE SERVICE call.

Application Data

For capture points that pass data between the application and CICS you may specify application data as the information source. For example, at the program initiation capture point, the choices for the application data areas are COMMAREA and CHANNEL. In this case CICS does not know the format and length of the data area so you must supply it.

For example, if you choose CHANNEL, you must supply the name of the container holding the data, the type of data, its offset into the container, and the length of the data. If the data type is Packed Decimal or Zoned Decimal, supply the precision. If the data type is Character, you can choose an alternate code page from the default code page IBM037. You can import a language structure for the application data to help specify this information.

Figure 2-11 shows the information source editor.

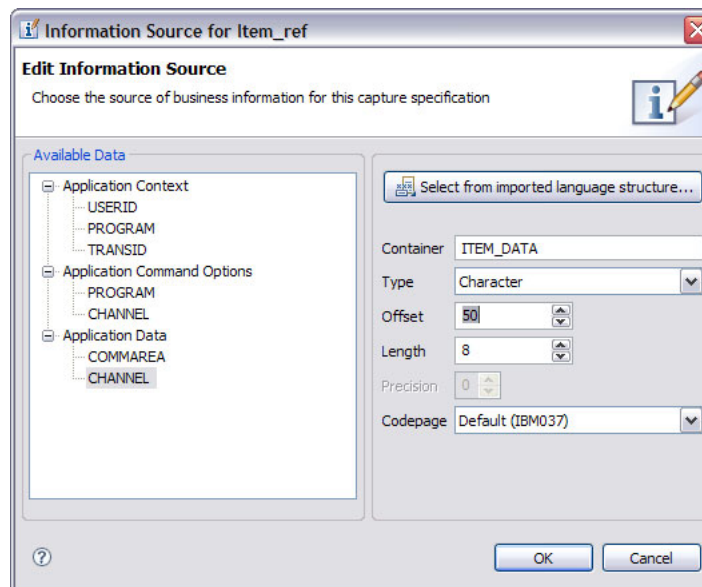


Figure 2-11 Editing an information source.

2.6 Event binding

An event binding is an XML definition that defines one or more business events to CICS. An event binding consists of event specifications, capture specifications, and adapter and dispatcher information.

The event binding is the unit for installing, enabling, and disabling CICS events. The event binding groups together sets of events that are to be handled using the same EP adapter, configuration, and dispatching policy. Event bindings are deployed to CICS in a bundle that can contain other resources. All of the resources in a bundle can be enabled and disabled together.

You create an event binding in a CICS bundle project using the CICS event binding editor. Figure 2-12 shows an event binding containing three event specifications.

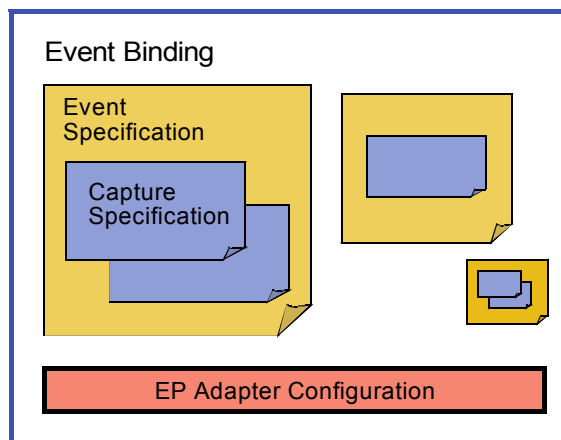


Figure 2-12 Event Binding containing 3 event specifications.

The event binding contains the following information:

- Description
- User tag

Tip: Use the CICS Explorer CICS SM perspective to display the user tag for an event binding installed in CICS.

Figure 2-13 shows the event binding details for the catalog sample. The EP adapter information is shown on the Adapter tab.

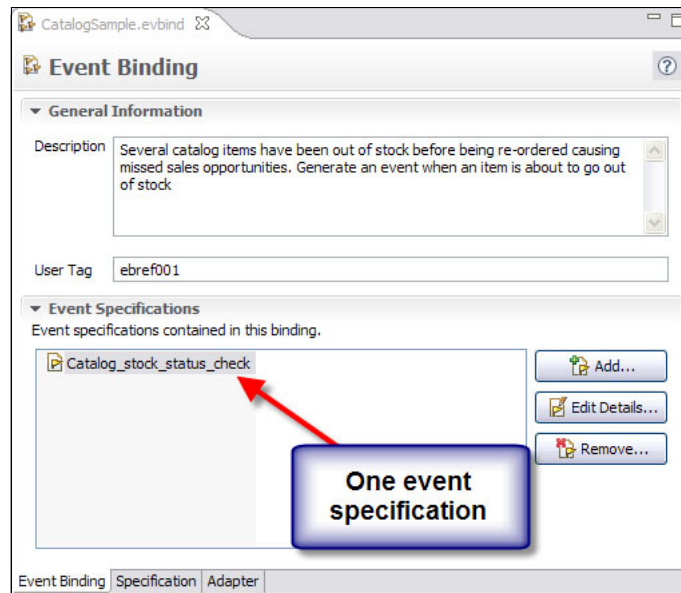


Figure 2-13 Catalog sample event binding

2.7 Non-invasive events or SIGNAL EVENT

As described above CICS EP support allows existing business applications to be instrumented to emit events without altering the existing application code. We call this non-invasive event processing.

If you wish to identify explicitly a place in an application program where one or more events could be emitted, add an EXEC CICS SIGNAL EVENT call to your program. This is invasively adding the opportunity for an event to be captured. The SIGNAL EVENT call is also useful because you might need to collect the data available for the event payload from diverse sources. You can collect the information in containers on the FROMCHANNEL or pass a single data area using FROM.

The SIGNAL EVENT identifies a place in an application program where one or more events could be emitted. Events are emitted when the following conditions are satisfied:

- ▶ Event processing is active.
- ▶ There is at least one matching capture specification enabled. A capture specification matches if it has a capture point of SIGNAL EVENT, and all its predicates evaluate to TRUE.

SIGNAL EVENT has a primary predicate of EVENT, and allows other predicates on the FROM data-area or the FROMCHANNEL and its containers. You can also define predicates on the context data, as for all capture points. The data in any CICS Event emitted as a result of SIGNAL EVENT is defined in the business event that contains the matching capture specification.

SIGNAL EVENT works in the same way as any other capture point in CICS. The differences are as follows:

- ▶ You can explicitly choose where in the application logic the capture point occurs
The capture points we provide may not meet your requirements.
- ▶ You provide the name for this capture point.
 - This might be unique or provide a common name, such as the application area.
 - Capture points can filter on this name with Equals or Begins With, for example.
- ▶ You may assemble the data you wish to make available from diverse sources.
A capture specification for this signal event can then pick information sources from this data to build the event payload

SIGNAL EVENT is also useful for application vendors to provide pre-prepared capture points for their customers. Again, a range of useful information may be provided for event payloads.

2.7.1 Automatic Capture Specification for SIGNAL EVENT

If you create a SIGNAL EVENT using FROMCHANNEL and write the data into containers, you can use the editor to create a capture specification for the event automatically.

You add an automatic capture specification using the event binding editor, by performing the following steps:

1. Add a new event specification with a name which matches the EVENT predicate.
2. Add the business data you wish to capture with the names of the containers you are passing on the FROMCHANNEL.
3. Press Add an automatic capture specification button.

The editor creates a capture specification for a SIGNAL EVENT capture point with the filter as follows:

```
EVENT Equals <name of event specification>
```

It also adds information sources for each business data item, as follows:

- ▶ The information is in a container with the name of the business data passed on the FROMCHANNEL
- ▶ The data is at offset 0 in the container
- ▶ The length is the same as the length of the business data

Figure 2-14 shows an event specification to be used with SIGNAL event.

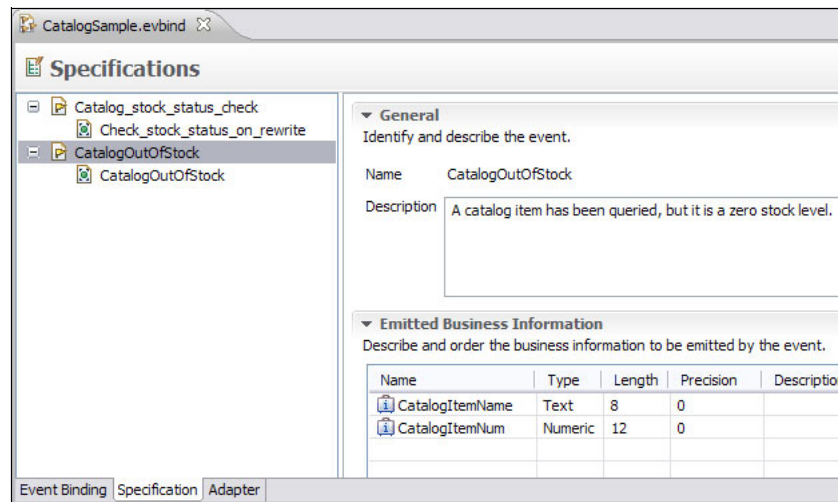


Figure 2-14 Event specification for a SIGNAL EVENT

Figure 2-15 on page 31 shows an automatically-generated capture specification.

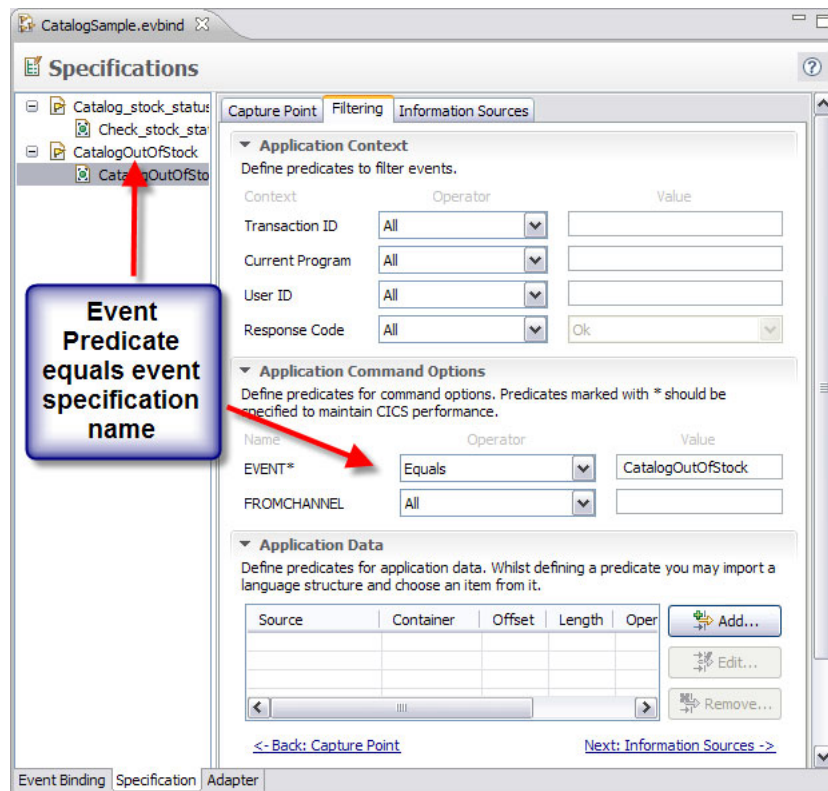


Figure 2-15 Automatically generated capture specification

Figure 2-16 shows the automatically-generated information sources in the capture specification.

Name	Type	F...	Source	Container	Offset	Capture Length	Capture Type
Program...	Text	8	FROMCHANNEL	Program_name	0	8	Character
Item_ref	Numeric	4	FROMCHANNEL	Item_ref	0	4	Character
Item_des...	Text	40	FROMCHANNEL	Item_description	0	40	Character
in_stock	Numeric	4	FROMCHANNEL	in_stock	0	4	Character
on_order	Numeric	3	FROMCHANNEL	on_order	0	3	Character

Figure 2-16 Automatically generated information sources

2.8 Event Processing Adapters

Specify information in your event binding to control how CICS emits events produced by the event binding. Use the adapter panel to define what will happen to events created by this binding. Select the EP adapter to emit events, then select options relevant to the EP adapter. Figure 2-17 on page 33 shows the adapter configuration for the catalog sample.

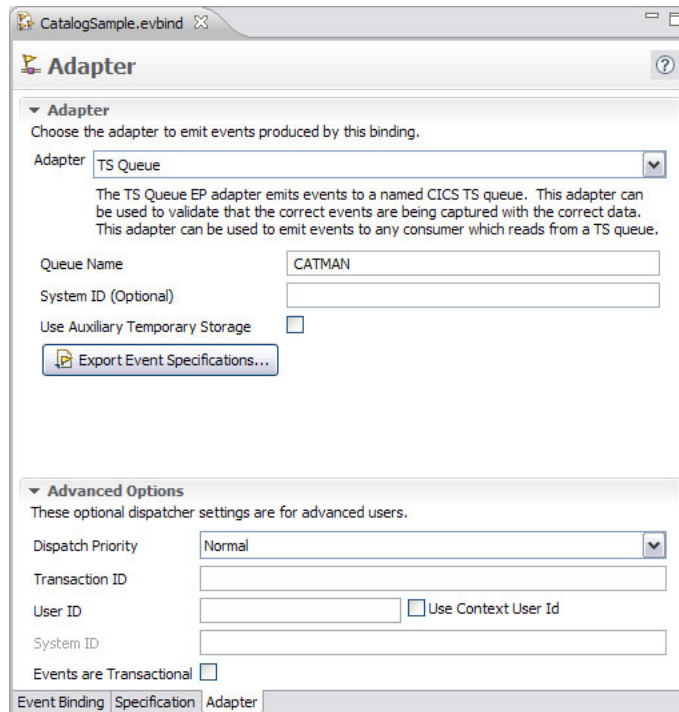


Figure 2-17 Adapter information for the catalog sample

Click the **Adapter** tab in the CICS event binding editor as shown. The adapter pane is where you specify the type of EP adapter to use for this event binding, the parameters for the EP adapter, and any advanced information.

Choose the **EP adapter** type from the Adapter list. You can specify four EP adapter types, discussed in 2.8.1, “TS Queue EP adapter” on page 34 through 2.8.3, “WMQ queue EP adapter” on page 35

You may export a description of your event as an XML schema or COBOL copybook. See 2.9, “Exporting event schema or copybook” on page 36.

You can optionally set advanced dispatcher settings. See 2.10, “EP Adapter advanced options” on page 37.

2.8.1 TS Queue EP adapter

This EP adapter emits events in the CICS flattened event (CFE) format to a named CICS temporary storage queue and can be used to perform the following tasks:

- ▶ Validate that the correct events are being captured with the correct data.
- ▶ Emit events to any consumer that reads from a TS queue.

This EP adapter is a good choice when implementing events. You can use it as a temporary adapter because you can browse TS Queues with the CICS CEBR transaction easily to check the payload. You can refresh the view to check for new events in the queue.

However, TS Queues are limited to 32,767 entries or less. You may only delete the entire queue rather than individual entries within, so you will need a strategy for clearing out events you have processed.

Specify the following options for the TS Queue EP adapter:

- ▶ Specify the CICS queue name. You must specify a queue name.
- ▶ Specify the System ID, if your target queue is remote.
- ▶ Select **Use Auxiliary Temporary Storage** if required.

2.8.2 Transaction start EP adapter

This EP adapter emits events in the CICS container-based event (CCE) format to a named CICS transaction that consumes the event. You can specify the CICS system that will run the transaction. You can use an existing transaction, if the event data is not required.

Specify the following options for the Transaction start EP adapter:

- ▶ Specify the transaction ID of the CICS application that runs as a result of the events. You must specify a transaction ID.
- ▶ Optionally, specify a transaction user ID.

If unspecified, the started transaction will run under the CICS region user ID.

The transaction that is started by the transaction start EP adapter will run under this user ID. The CICS region user ID needs to be defined as a surrogate of any user ID specified here.

2.8.3 WMQ queue EP adapter

This EP adapter emits events to a WebSphere MQ queue either in an XML format for consumption by products that use the common event infrastructure such as WebSphere Business Monitor, the common base event (CBE) format for WebSphere Business Monitor, or in a non-XML character format.

Specify the following options for the WMQ Queue EP adapter:

- ▶ Specify the Queue Name of the WebSphere MQ queue on which events emitted by this event binding are placed. You must specify a queue name.

Specify whether messages are persistent. Select one of the following values from the Persistent list.
 - No
Messages put on the queue by the adapter are nonpersistent.
 - Yes
Messages put on the queue by the adapter are persistent.
- ▶ Queue default
Messages put on the queue inherit the default persistence of the named queue.
- ▶ Specify the message priority. You can either select the queue default, or type a value in the Priority field, for the WebSphere MQ message priority, from 0–9.
- ▶ Specify the expiry time. You can either select Never Expire, or type a value for the WebSphere MQ message expiry in the Expiry Time field. This period of time is expressed in tenths of a second. A message becomes eligible to be discarded if it has not been removed from the destination queue before this period of time elapses.
- ▶ Specify a data format for the event. Select one of the following values from the Data Format list.
 - CICS Flattened Event (CFE) format
Event data is in a non-XML character format.
 - WebSphere Business Events XML format
Messages are put on the queue in the XML format required by WebSphere Business Events.
 - CBE format for WebSphere Monitor
Messages are put on the queue in the CBE event format required by WebSphere Business Monitor.

2.8.4 Custom (user written) EP adapter

This adapter emits events in any format that you require. A custom EP adapter is a CICS program that you write to provide a combination of formatting and routing of an event that is not supported by the CICS-provided EP adapters. It must not carry out any other processing, such as consumption of the event.

Specify the transaction ID for your user-written CICS application that formats, routes, and emits the event. You must specify a transaction ID.

Write the data to be passed to the Custom EP adapter. Your Custom EP adapter will receive this data which may be used to configure the custom EP adapter

2.9 Exporting event schema or copybook

You can export descriptions for one or more event specifications in this event binding as a schema or copybook for use elsewhere.

If you are using the WMQ Queue adapter and the CBE format or the WebSphere Business Events (XML) format, the exported file will be an XML schema definition (.xsd) file.

If your chosen adapter and data format emit events to a system in a non-XML character format, the exported file will be a COBOL copybook (.cpy) file. The TS Queue adapter and the WMQ Queue adapter using the CICS Flattened Event format emit events in a non-XML character format.

To export event schema or copybook, perform the following steps:

1. Click **Export Event Specifications**. The “Export Event Specifications” window is displayed.
2. Select the event specifications that you want to export
3. Specify a directory to which to export the event specifications in the “To” directory field.
4. Click **Export**.

A file is created in the specified directory for each event specification that you selected.

For example, if you specify the WMQ Queue adapter and the CBE format and select two event specifications called “example1” and “example2”, two XML schema files are created, `example1.xsd` and `example2.xsd`. You can import this schema file to WebSphere Business Monitor to help define an inbound event.

If you specify the WMQ Queue adapter and the WebSphere Business Events (XML) format and select an event specification called “example1”, an XML schema file is created, called `example1.xsd`. You can use this schema file in the WebSphere Business Events Design Data tool to help define an event.

If you specify the TS Queue adapter and select two event specifications called “example1” and “example2”, two COBOL copybooks are created, `example1.cpy` and `example2.cpy`. You can use these copybooks to process data in your own event consumer programs.

If you export an event specification, and then export the same event specification again to the same directory, the CICS event binding editor prompts you to either overwrite the existing file or cancel the export operation.

2.10 EP Adapter advanced options

Specify any required advanced dispatcher options shown Figure 2-18. These options are for advanced users, and control the way in which the EP adapter is run in a CICS system. They are not required.

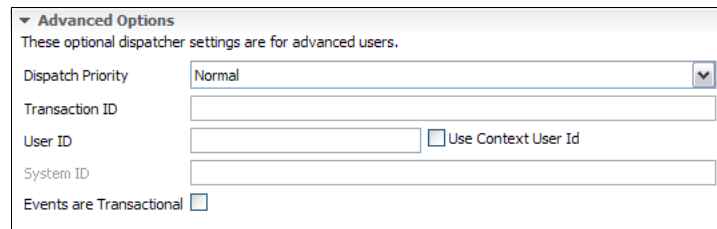


Figure 2-18 Advanced options for dispatch

2.10.1 Dispatch priority

You can specify Normal or High priority to control dispatching of events associated with this event binding. High priority events are emitted as soon as they are available based on the **Events are Transactional** setting. Normal priority events are emitted as soon as they are available, unless **Events are Transactional** is set, but after any outstanding high priority events.

2.10.2 Transaction ID

Specify the Transaction ID (not available for the CICS Transaction EP adapter, or a custom EP adapter). The EP adapter program will run under this transaction ID. If you do not specify a transaction ID, and you do not specify a user ID, the EP adapter is linked to under the dispatcher transaction.

2.10.3 User ID

You specify a user ID so that the EP adapter transaction runs with this user ID otherwise it runs under the CICS region user ID. If you select **Use context User ID**, the EP adapter runs with the user ID under which the event was captured.

If you specify a user ID, but you do not specify a transaction ID, the EP adapter will run under the default transaction for the EP adapter type:

- ▶ The WMQ Queue EP adapter runs under the CEPQ transaction.
- ▶ The TS Queue EP adapter runs under the CEPT transaction.

When you install a BUNDLE resource that includes an event binding for which you specified a user ID in the Adapter tab of the CICS event binding editor, CICS checks that the User ID performing the install operation is authorized as a surrogate user of the user ID specified in the CICS event binding editor. This check also applies to the CICS region user ID during group list install on a CICS cold or initial start.

2.10.4 System ID

Specify the system ID (only available for the CICS Transaction EP adapter). The transaction started by the EP adapter will run on the CICS system with this system ID.

2.10.5 Transactional events

Specify whether events are transactional.

- ▶ Select the check box if you want CICS to emit captured events only if the unit of work (UOW) associated with the event completes successfully. The events are either emitted when there is an internal sync point within the program, or when the program ends and reaches an implicit sync point.

If the transaction fails and backs out, the events will be discarded.

- ▶ Clear the check box if you want CICS to process events associated with this event binding in near-real-time regardless of whether the UOW commits.

2.11 Bundles

You can now deploy applications into CICS using bundles. A bundle is a collection of CICS resources, artifacts, references, and a manifest that represent an application. Use bundles to more easily manage the availability of an application and the life cycle of its resources.

For an example of a bundle containing three event bindings, see Figure 2-19.

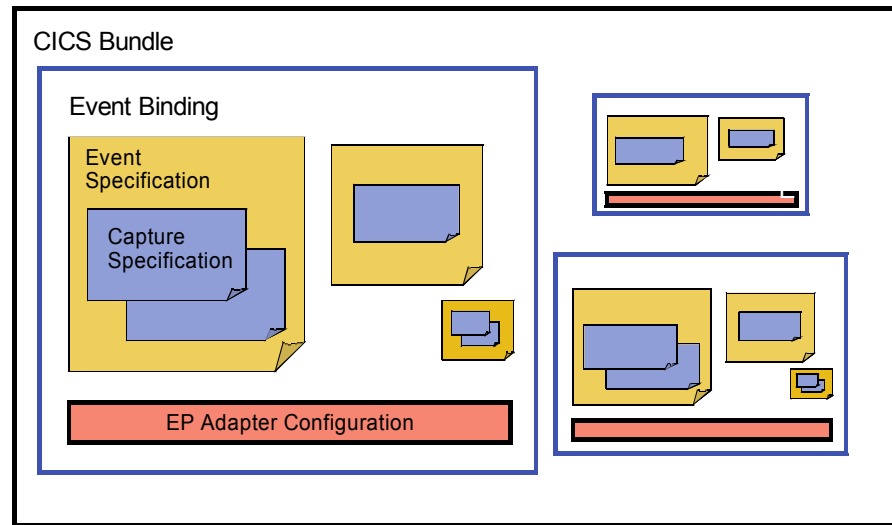


Figure 2-19 A CICS bundle containing three event bindings.

Bundles are created by an application developer using a tool such as Rational Developer for System z, the CICS XML assistant, or the IBM CICS Explorer. A bundle contains only the resources that are required by the application. The system resources that the application requires might be defined as prerequisites, but they are not included in the bundle. This separation means that you can install the same application into multiple CICS regions without having to repack or redeploy the bundle.

A bundle is defined in CICS using a BUNDLE resource. For information about how to define this resource and for more information about the format of its contents, see the following Web page:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/topic/com.ibm.cics.ts.resourcedefinition.doc/bundles/defining_app_resources.html

The BUNDLE resource is different from a Resource Definition Online (RDO) group because it maintains a relationship with all the resources after they are installed, so that you can manage all the related resources as a single entity. For example, if you disable a BUNDLE resource because you want to stop an application from running, CICS disables all of the related application resources for you. To view the contents of a bundle and the state of its resources, use the IBM CICS Explorer.

The types of applications that you can deploy as bundles include event processing, channel-based services, and XML-based services. Each of these application types is represented by one or more CICS resources. These resources are dynamically created as part of the bundle deployment.

If you deploy an application that uses event bindings from the CICS event binding editor, installing the BUNDLE resource generates one or more EVENTBINDING and CAPTURESPEC resources. The resource signatures for each resource indicate that they were created during a bundle deployment and contain the name of the BUNDLE resource.

2.12 Deploy a bundle to zFS

You export the bundle to z/OS UNIX® (zFS) storage from where it can be installed into CICS by referencing the bundle directory in a CICS bundle resource definition. To deploy a bundle to zFS, right-click the bundle and select the **deploy** option, as shown in Figure 2-20.

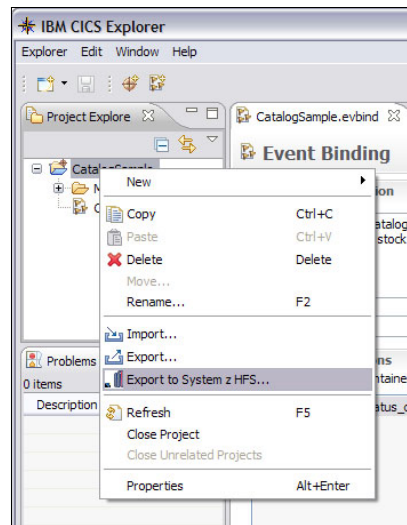


Figure 2-20 Selecting the export panel

Complete the information in the Export panel to provide the host, FTP port, username, and password for the zOS system. Enter the HFS directory or browse for the desired directory. See Figure 2-21.

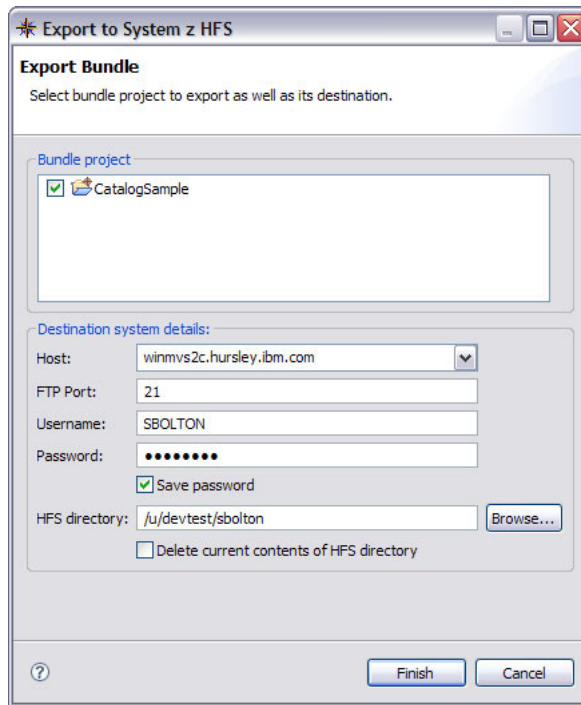


Figure 2-21 Export bundle panel



Part 2

Implementing CICS event processing

This part of the book focuses on implementation of event processing in CICS. It includes a step-by-step guide to implementing CICS event processing, along with a look at our environment, plus a quick look at some troubleshooting ideas.



Environment overview

In this chapter we describe the environment and the shopping sample application we use in the following chapters:

- ▶ Chapter 7, “WebSphere Business Events scenario” on page 183.
- ▶ Chapter 8, “WebSphere Business Monitor scenario” on page 213.

3.1 Example environment

To provide examples and demonstrate how to use event processing, a basic environment was created. See Figure 3-1 which shows the structure of the environment. If you wish to follow the examples given you will need to have access to an environment with similar components, however the structure is flexible.

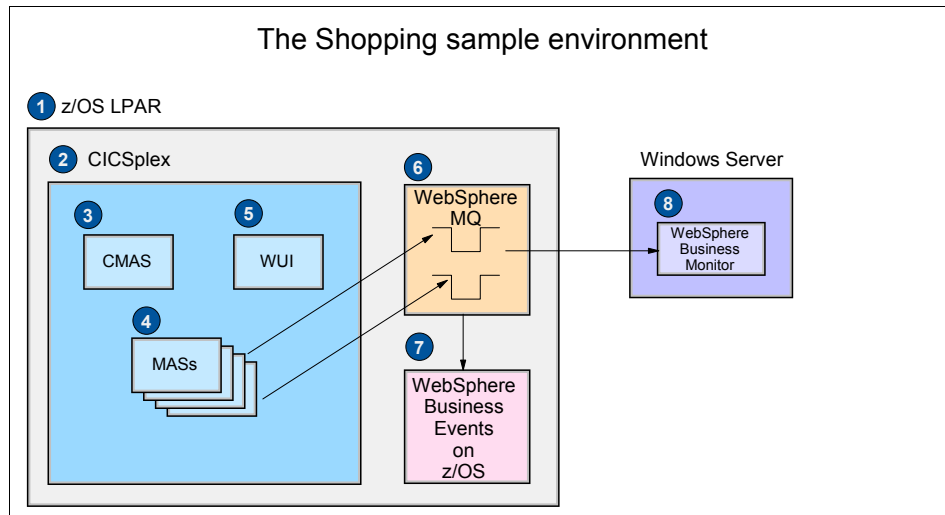


Figure 3-1 The environment we use for the shopping sample application

The components that are indicated in Figure 3-1 are as follows:

1. The example environment existed on a single z/OS LPAR.
2. A single CICSplex was used to enable CICS Explorer to connect through a WUI and provide update facilities.
3. The CICSplex requires a maintenance point CMAS.
4. The CICSplex has multiple MASs in which the example application runs and events are emitted
5. A WUI is required to allow the CICS Explorer to connect to the CICSplex. To allow the update functions of CICS Explorer the CMCIPORT(99999) parameter needs to be defined.
6. A WebSphere MQ Queue Manager is required to allow events to be emitted from the MASs to both WebSphere Business Monitor (WBM) and WebSphere Business Events (WBE). Within the queue manager, separate queues will need to be defined to transport the events to either WBE or WBM.

7. The environment we used to develop the examples had WebSphere Business Events for z/OS running on the same z/OS LPAR as our CICS regions. However, in your environment you can use any platform supported by WBE.
8. The example environment has WebSphere Business Monitor installed on a Microsoft® Windows Server®.

3.2 Shopping sample application

To provide the examples, we developed an simple order processing application. This application is not intended to represent a real life application, it is intended to allow multiple opportunities for event processing API capture points. Figure 3-2 shows the order process that the example application implements.

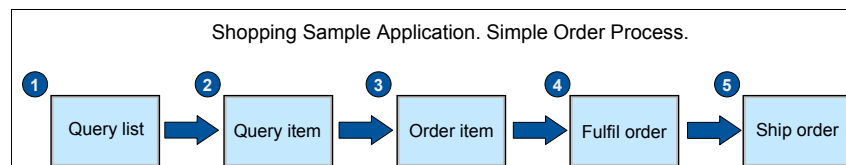


Figure 3-2 Shopping sample application, order process

The stages of the order process are as follows:

1. A customer displays a list of stock items on the panel.
2. A stock item is displayed showing the stock level and price.
3. If satisfied with the selection, the stock item is ordered and the order is posted to the ORDER file.
4. The warehouse takes orders and fulfills them by reducing the stock levels.
5. The warehouse marks the order as shipped.

A set of programs, transactions, BMS maps, and copybooks have been developed to implement the above order process. The structure of the application is shown in Figure 3-3 on page 48,

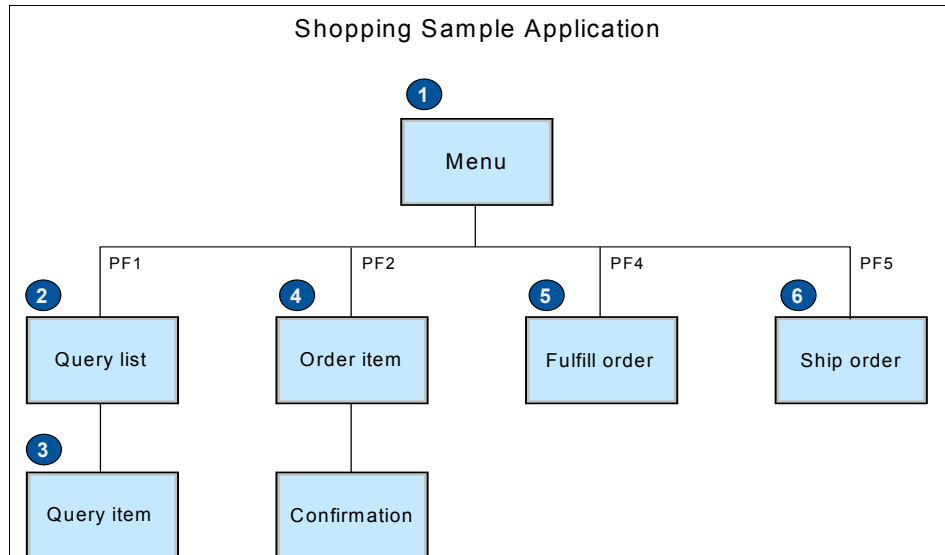


Figure 3-3 Shopping sample application structure

The components of the Shopping sample application shown in Figure 3-3 are as follows:

1. Transaction and program MENU provides the ability to drive the various functions of the application. The BMS map (MENU1) has a field to enter a five digit customer number, required for both the query list and order item functions.
2. From the menu, providing a customer number and pressing PF1 starts the psuedo-conversational transaction QRY and program QUERY. This initially brings up a list of stock items from the STOCK file.
3. Using any character to select a stock item from the list, displays the stock level and price of the item.
4. From the menu, providing a customer number and pressing PF2 starts the conversational program ORDER. This allows the users to enter the stock item ID and quantity to order. Program SENDORDR is called to generate a record on the ORDER file and a confirmation panel is displayed.
5. From the menu, pressing PF4 will simulate the warehouse fulfilling all outstanding orders by running program fulfill. A message will be displayed on the menu detailing how many items have been fulfill and the total value of all the orders.
6. From the menu, pressing PF5 will simulate the warehouse shipping all fulfilled orders by running program SHIP. A message will be displayed on the menu detailing how many items have been shipped.

The shopping sample application with all its source and setup JCL can be downloaded from the Redbooks website. The provided `readme.txt` file details how to upload the files to z/OS, configure and run the setup JCL.

3.3 Sample scenarios

To allow the following chapters to provide semi-realistic demonstrations, a set of four scenarios have been developed to be used with the example application. These are:

- ▶ Query versus sale
- ▶ Stock low
- ▶ Shipped order meets SLA
- ▶ High-value order breakdown

3.3.1 Scenario: Query versus sale

This scenario will e-mail a discount offer to a customer if the customer has queried a stock item repeatedly (four times), but not ordered it. The theory behind this is the customer is interested in an item but may need a little encouragement to complete the sale. Figure 3-4 shows the flow of the scenario.

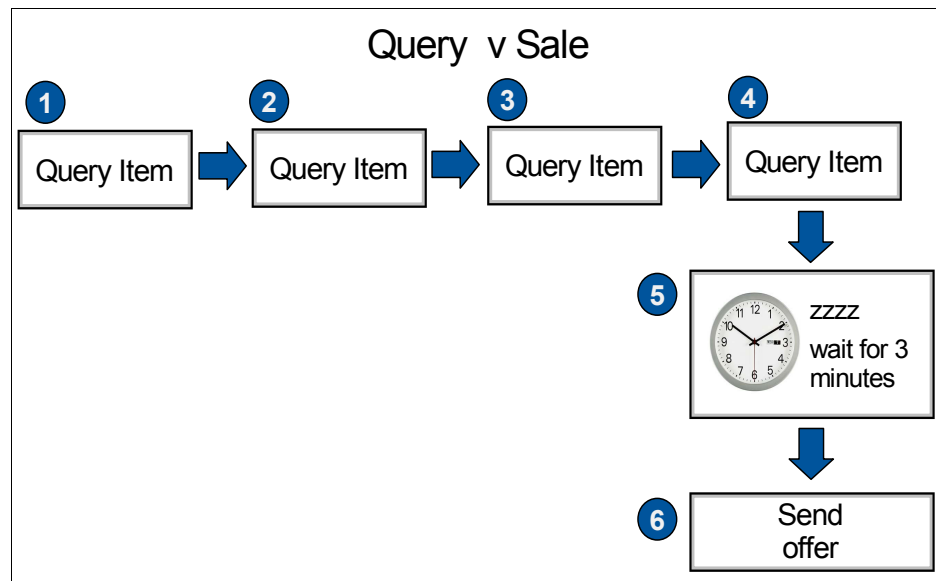


Figure 3-4 Query versus sale scenario

The stages of the scenario shown in Figure 3-4 on page 49 are as follows:

1. The customer queries the item for the first time.
2. The customer repeats the query.
3. The customer repeats again.
4. The customer repeats the query for the fourth time.
5. The scenario will wait for a period of time (in the examples, this is three minutes for demonstration purposes).
6. If an order with the same stock ID and customer number has not been received in that time, an e-mail will be sent with a discount offer.

For this scenario to work, two events will need to be emitted from the example application, the first event will be emitted whenever a query item is performed and the second whenever an order is placed.

This type of scenario is best suited for WebSphere Business Events to process the events emitted from CICS TS. Therefore, the WMQ Queue EP adapter using would be used to send events from CICS TS to the WBE server.

3.3.2 Scenario: Stock low

This scenario will send an event whenever an order is fulfilled and the stock level is below 50 units. The theory behind this is to alert the warehouse through a dashboard that the item will need to restocked. Figure 3-5 shows the flow of the scenario.

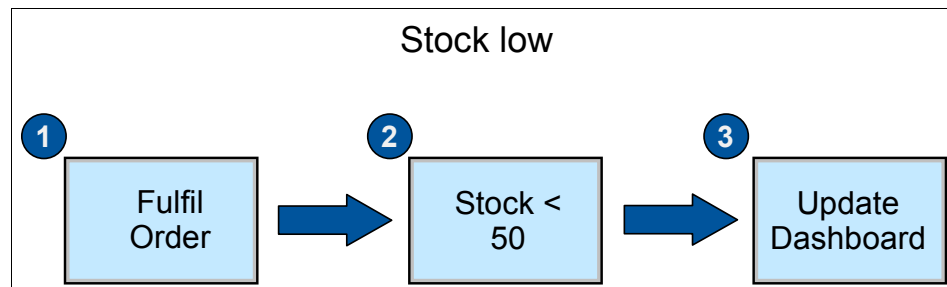


Figure 3-5 Stock low scenario

The stages of the scenario shown in Figure 3-5 are as follows:

1. The fulfill process allocates stock to each outstanding order.
2. The stock level is under 50.
3. Update a dashboard with a request to replenish the stock.

For this scenario to work, an event needs to be emitted whenever the stock record is updated and the level is below 50.

This type of scenario is best suited for WebSphere Business Monitor to process the events emitted from CICS TS and will use the WMQ Queue EP Adapter.

3.3.3 Scenario: Shipped order meets service level agreements

This scenario will check that the time between an order being placed and shipped is within service level agreement requirements. This scenario can be used to monitor the warehouse's efficiency. Figure 3-6 shows the flow of the scenario.

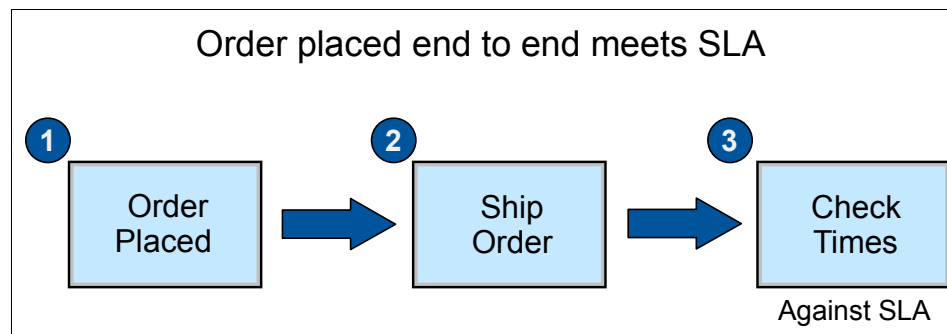


Figure 3-6 Order end to end scenario

The stages of the scenario shown in Figure 3-6 are as follows:

1. An order is placed.
2. The warehouse ships the order.
3. The times are compared and a Key Performance Indicator is updated to reflect the warehouse's efficiency.

Events are to be emitted whenever an order is placed and shipped. The order placed event used in 3.3.1, "Scenario: Query versus sale" on page 49 can be reused here.

This type of scenario is best suited for WebSphere Business Monitor to process the events emitted from CICS TS and will use the WMQ Queue EP Adapter using CBE format.

3.3.4 Scenario: High-value order breakdown

This scenario will allow a dashboard to present a breakdown of what high-value orders are being placed within the system. The dashboard can be configured to present order distribution based on demographics, customer importance, order values and hot stock items. Figure 3-7 shows the flow of the scenario.

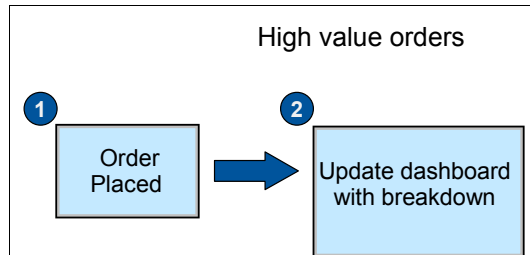


Figure 3-7 High value order scenario

The stages of the scenario shown in Figure 3-7 are as follows:

1. An order is placed.
2. The dashboards are updated with the new breakdowns.

An event will be emitted whenever an order is placed. As the amount of data required to be emitted with the event is more than can normally be available at a standard capture point, this scenario will need to be an intrusive change to the application code so that all the data can be collated and emitted together.

This type of scenario is best suited for WebSphere Business Monitor to process the events emitted from CICS TS and will use the WMQ Queue EP Adapter.

3.4 The events emitted in the scenarios

The capture points in the examples are based on the CICS API calls made in each phase of the order process in the shopping sample application (Query stock, Order Item, fulfill Order, and Ship Order). Table 3-1 on page 53 shows the information contained in the capture specifications for our sample event binding. The 'Emitted Business Information' column contains the payload data captured with the events, which is selected from the containers passed with each API command.

Table 3-1 The capture points in the sample MENU application

Event Name	Capture Point	Predicates	Emitted Business Information
QueryStock	Program Initiation	Program Name = QUERY	CustomerNumber: Numeric Packed-Decimal Capture Length = 3 Format Length = 6 StockId: Numeric Packed-Decimal Capture Length = 3 Format Length = 6
Order	Link Program (Capture after)	Program Name = SENDORDR	CustomerNumber: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6 OrderNumber: Numeric Packed-Decimal Capture Length = 3 Format Length = 6 StockId: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6 Quantity: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6
fulfill	Put Container	Program name = UPDSTOCK Container name = OUTPUT Stock item level is < 50	StockId: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6 OldLevel: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6 NewLevel: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6

Event Name	Capture Point	Predicates	Emitted Business Information
Ship	Rewrite	Program name = SHIP Response Code = Ok File name = ORDER Order status = 'S'	OrderNumber: Numeric Packed-Decimal Capture Length = 3 Format Length = 6 CustomerNumber: Numeric Packed-Decimal Capture Length = 3 Format Length = 6 StockId: Numeric packed-Decimal Capture Length = 3 Format Length = 6 Quantity: Numeric Zoned-Decimal Capture Length = 5 Format length = 6

The capture points are based on the API calls in the shopping sample application programs, as shown in Table 3-2.

Table 3-2 API commands in the MENU sample application, on which capture points are based

Event Name	Capture Point	MENU Sample program Name	API command
QueryStock	Program Initiation	QUERY	EXEC CICS RETURN TRANSID('QRY') CHANNEL('LIST-STOCK') END-EXEC.
Order	Link Program (Capture after)	SENDORDR	EXEC CICS LINK PROGRAM('SENDORDR') CHANNEL(CHANNEL-NAME) END-EXEC.
fulfill	Put Container	UPDSTOCK	EXEC CICS PUT CONTAINER('OUTPUT') FROM(UPDATE-OUT) END-EXEC.
Ship	Rewrite	SHIP	EXEC CICS REWRITE FILE('ORDER') FROM(ORDER-REC) END-EXEC.

In addition to the above non-invasive capture points, we also included the following EXEC CICS SIGNAL EVENT API call in the SENDORDR sample program to demonstrate how this technique of capturing events could be used if you have the opportunity to alter your source code to capture events:

```
EXEC CICS SIGNAL EVENT('SendOrder')
  FROM(SIGNAL-EVENT-DATA)
  FROMLENGTH(LENGTH OF SIGNAL-EVENT-DATA)
END-EXEC.
```

The SIGNAL EVENT details are shown in Table 3-3.

Table 3-3 SIGNAL EVENT capture point in Shopping sample application

Event Name	Capture Point	Predicate	Emitted Business Information
SendOrder	Signal Event	Event name = SendOrder	CustomerNumber: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6 OrderNumber: Numeric Zoned-Decimal Capture Length = 5 Format Length = 6 CustomerName: Text Character Capture Length = 50 Format Length = 50 City: Text Character Capture Length = 50 Format Length = 50 Country: Text Character Capture Length = 50 Format Length = 50 Premium: Text Character Capture Length = 1 Format Length = 1 TotalOrderValue: Numeric Zoned-Decimal Capture Length = 11 Format Length = 13 Precision = 2



Setting up CICS for events

In this chapter we look at how we set up CICS for events. We show how we provide the development tooling environment needed to create an event binding. We configure CICS TS V4.1 for event processing and use the CICS Explorer to control event processing in CICS. Working on the basis that we have been supplied with a bundle project containing an event binding, we show how we deploy and install this bundle into CICS. We show how we can disable and discard a bundle or event binding. We look at the CICS security options that relate to CICS event processing and show an example of how we protect access to bundles and event bindings using an external security manager, in our case, RACF®.

4.1 Our software versions

We used the following software versions in this chapter:

- ▶ CICS TS V4.1
- ▶ WebSphere MQ (WMQ) V6.0
- ▶ z/OS V1.9

Note: z/OS V1.9 is the minimum prerequisite for CICS TS V4.1

- ▶ CICS Explorer V1.0.0.1
- ▶ Windows XP SP2

4.2 CICS development setup

In this section we show how we obtain the CICS Explorer and connect it using the CICS management client interface (CMCI) to our CICSplex® SM Web User Interface.

The CMCI is a system management application programming interface, designed using representational state transfer (RESTful) principles, for use by HTTP client applications including IBM CICS Explorer.

We use the CICSplex SM Web User Interface because it allows us to use the CMCI to manage CICSplex SM managed resources.

Note: It is possible to connect the connect the CICS Explorer to use the single server (SMSS) version of the CICS management client interface in a stand alone CICS region without using CICSplex SM. Currently, when CMCI is set up in a CICS region, you can use it only to manage operational resources

For more information about using and connecting the CICS Explorer to stand-alone CICS region, see the CICS Infocenter at the following Web page:

https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/topic/com.ibm.cics.ts.clientapi.doc/topics/clientapi_setup.html

4.2.1 Obtaining the CICS Explorer

In this section we show how to configure CICS for event processing using the CICS Explorer. We downloaded the CICS Explorer onto to our mobile computer, which is running on the Windows XP platform. A Linux version of the CICS Explorer is also available. Download the CICS Explorer onto your workstation at the following CICS Explorer Web page:

<http://www-01.ibm.com/software/http/cics/explorer/>

With the CICS Explorer now running on your workstation, we show how to connect it to our CICS V4.1 CICSplex SM Web User Interface (EPREDW).

4.2.2 CICS Explorer connectivity

In CICS Explorer, click **Window Preferences** and select the connection settings field, as shown in Figure 4-1.

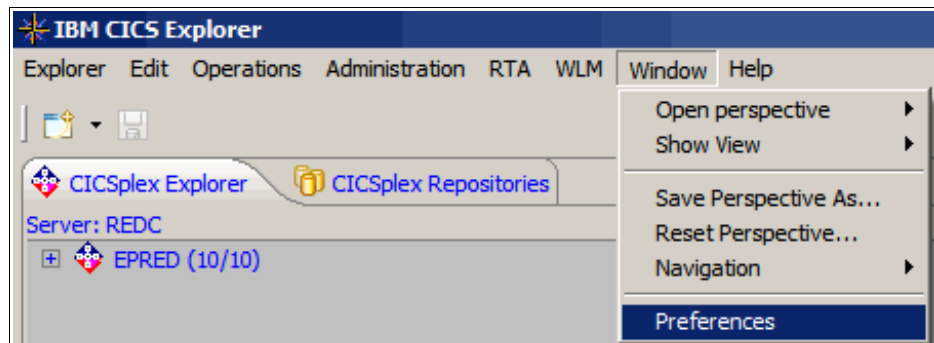


Figure 4-1 Explorer preferences

We used connection type “CICS management client interface (CMCI)” as shown in Figure 4-2.

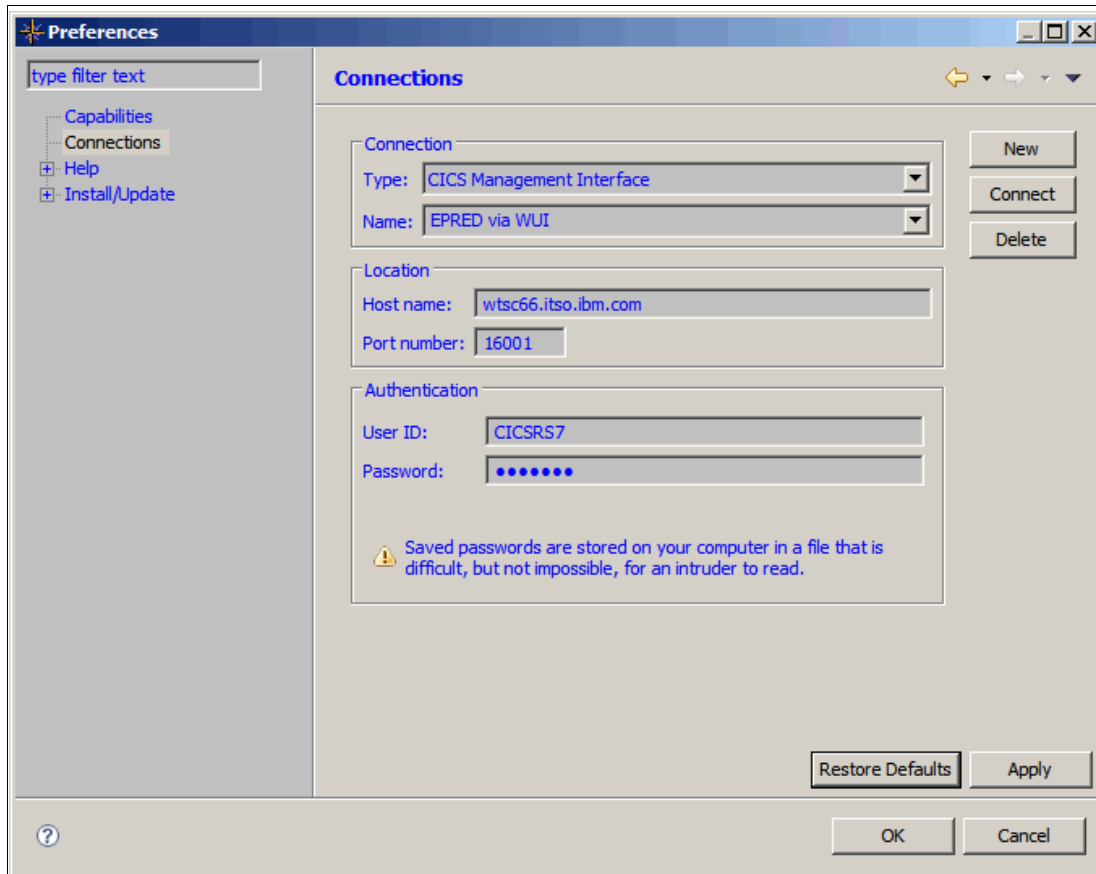


Figure 4-2 CICS Explorer connection options

Note: We use connection type “CICS Management Interface” to enable update capability. The default connection type “CICSplex SM Data Interface” allows read only capability.

Our WUI server (Hostname: wtsc66.itso.ibm.com) has port = 16001. The CMCI PORT system initialization parameter for our wui is set as 'CMCI PORT =16001'. See Figure 4-2.

See *Setting up the CICS management client interface* in the CICS Transaction Server for z/OS version 4 information center for further details on how this is done:

https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.clientapi.doc/topics/clientapi_setup.html

4.2.3 Obtaining the CICS event binding editor

The CICS event binding editor is supplied as part of the CICS Explorer which we downloaded in the previous section.

4.3 CICS System setup

In this section we will review the CICS system setup required to implement events in CICS.

4.3.1 Adding WebSphere MQ support to CICS region, to use the WMQ EP adapter

In our shopping application we use the WebSphere MQ EP adapter to format and process our events. We show the steps to add WebSphere MQ support to our CICS region. You might also need to configure WebSphere MQ (WMQ) if using the WMQ EP adapter.

Figure 4-3 shows how to add the WMQ loadlibs to the CICS STEPLIB and DFHRPL concatenations in our CICS startup JCL.

```
//STEPLIB DD DSN=CICSTS41.CPSM.SEYUAUTH,DISP=SHR
//        DD DSN=CICSTS41.CICS.SDFHAUTH,DISP=SHR
//        DD DSN=MQ600.SCSQANLE,DISP=SHR
//        DD DSN=MQ600.SCSQAUTH,DISP=SHR
//        DD DSN=CEE.SCEERUN,DISP=SHR
//        DD DSN=CEE.SCEERUN2,DISP=SHR
//DFHRPL  DD DSN=CICSTS41.CPSM.SEYULOAD,DISP=SHR
//        DD DSN=CICSTS41.CICS.SDFHLOAD,DISP=SHR
//        DD DSN=MQ600.SCSQANLE,DISP=SHR
//        DD DSN=MQ600.SCSQCICS,DISP=SHR
//        DD DSN=MQ600.SCSQLOAD,DISP=SHR
//        DD DSN=MQ600.SCSQAUTH,DISP=SHR
//        DD DSN=CEE.SCEECICS,DISP=SHR
//        DD DSN=CEE.SCEERUN,DISP=SHR
//        DD DSN=CEE.SCEERUN2,DISP=SHR
//        DD DSN=CICSSEM.LOADLIB,DISP=SHR
```

Figure 4-3 JCL showing WMQ loadlibs included

In the CICS SIT, because we have MQCONN=YES, the MQCONN resource will be installed at startup, as shown in Figure 4-4.

```
APPLID=EPRED7
SYSIDNT=RED7
START=INITIAL
GRPLIST=(DFHLIST,EPRED7)
MQCONN=YES
```

Figure 4-4 CICS SIT showing MQCONN

We use an RDO MQCONN definition as shown in Figure 4-5 on page 63.


```

OBJECT CHARACTERISTICS
CEDA View MQconn( MQ      )
MQconn      : MQ
Group       : MQRED7
DEScriptio  :
Mqname      : MQ8G
Resyncmember : Yes          Yes | No
Initqname   : EPRED7.TRIGGER
DEFINITION SIGNATURE
DEFinetime  : 06/25/09 10:24:49
CHANGETime  : 06/25/09 10:24:49
CHANGEUsrid  : CICSRS1
CHANGEAGEnt  : CSDBatch      CSDApi | CSDBatch
CHANGEAGRel  : 8884

                                SYSID=RED7 APPLID=EPRED7

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Figure 4-5 MQCONN definition

4.3.2 Enabling CICS event processing

Event processing is enabled by default when a START=INITIAL or START=COLD parameter is used during the startup of your CICS TS V4.1 systems.

When a START=WARM or START=EMERGENCY parameter is used, the settings from the previous run of CICS are used. So, unless the setting has been changed, event processing should be enabled. You can inquire on EVENTPROCESS to check this, for example by using the CICS Explorer.

You can stop and start event processing from the IBM CICS Explorer, the CICSplex SM Web User Interface, or the CICS SPI or API commands.

You might want to stop event processing for an upgrade or system maintenance, and then start event processing again.

CICS event processing can be set using three possible states: START, DRAIN, or STOP

DRAIN causes event capture to stop but allows events that have already been captured to be processed through the system and emitted. (Transactional events will not be emitted if the unit of work had not reached sync point at the time the DRAIN request is received) STOPPED causes event processing to stop immediately, with no further events being captured or emitted.

4.3.3 Stopping CICS event processing

Perform the following steps to stop event processing in the CICS region:

1. Using the CICSplex Explorer view, click the CICSplex or CICS region to select the CICSplex or CICS region on which you want to stop event processing.
2. Using the IBM CICS Explorer toolbar, click **Operations > Event Processing**. The current status of event processing is displayed. See Figure 4-6.

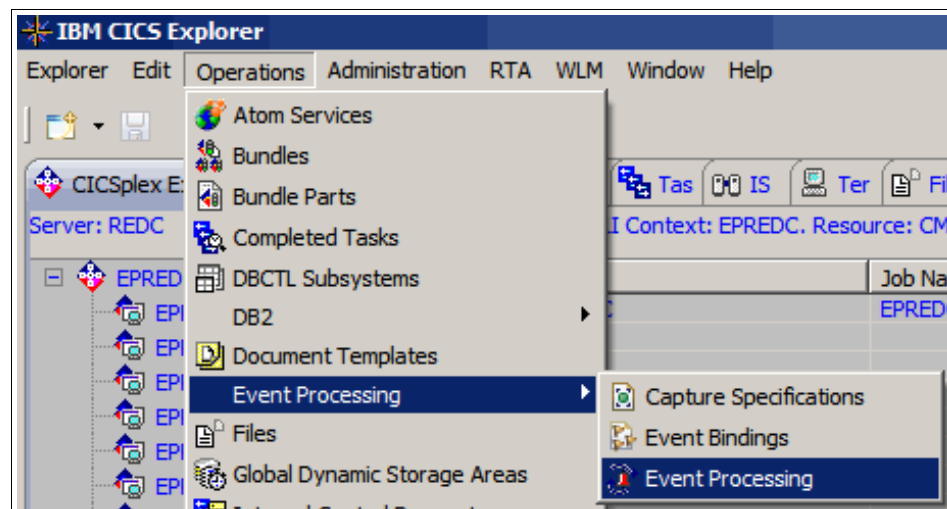


Figure 4-6 Controlling event processing

3. Right-click to select the region on which you want to stop event processing and click **Stop**. See Figure 4-7 on page 65.

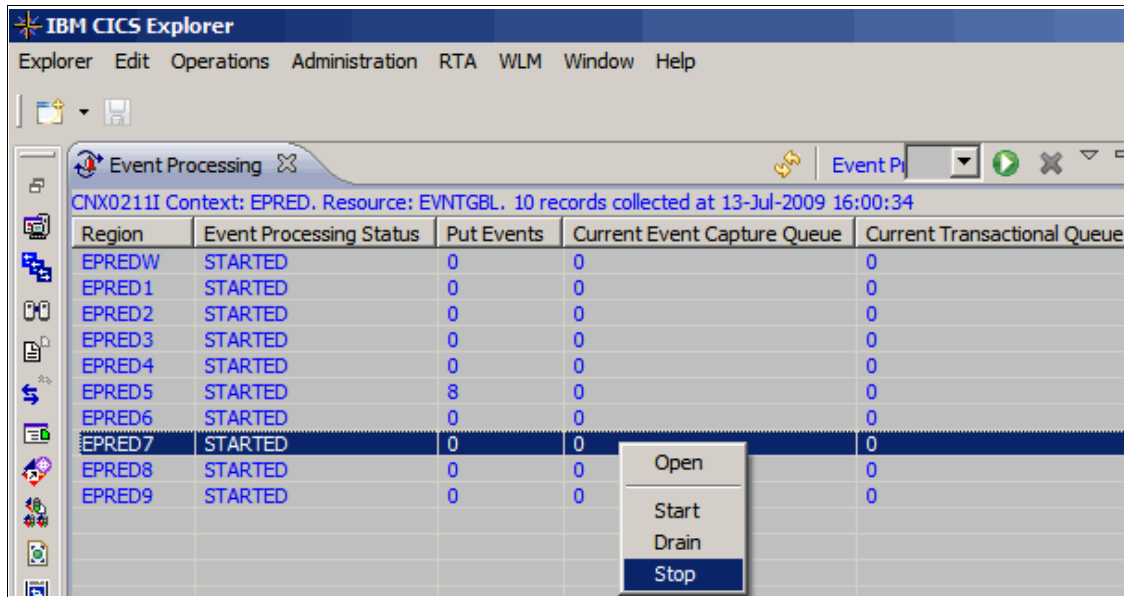


Figure 4-7 Stop event processing

You will get a warning panel to confirm event processing stop, as in Figure 4-8.

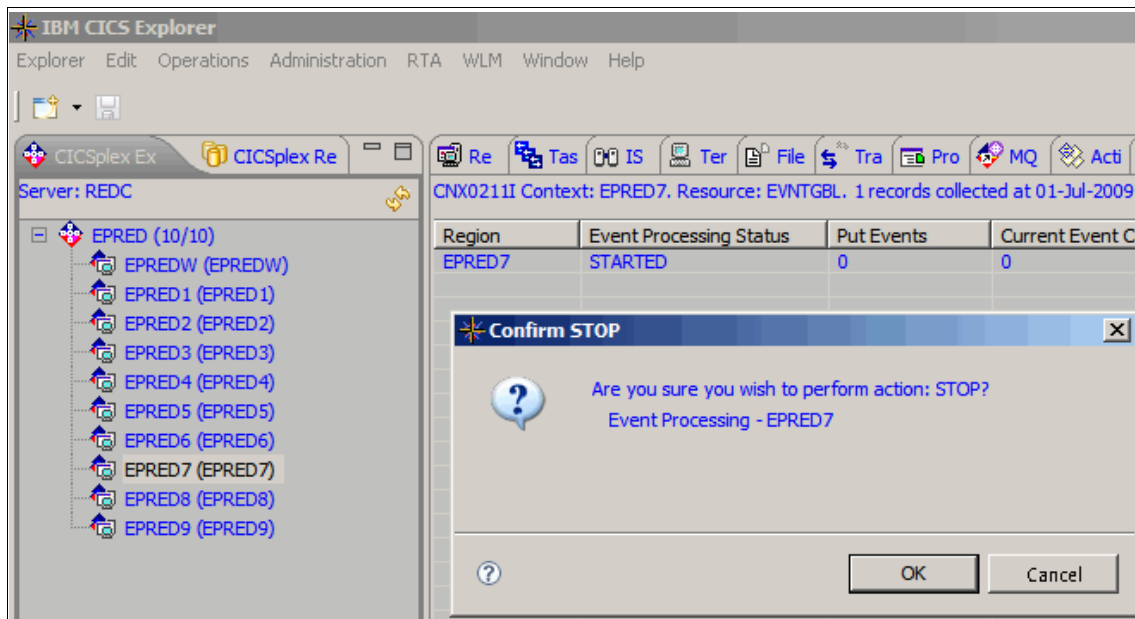
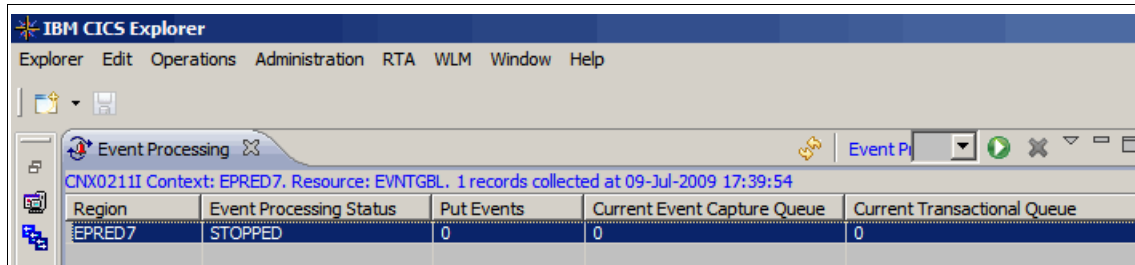


Figure 4-8 Event processing Warning

Note: The status will not update until you click the refresh icon. The STOPPED status displays, indicating that event processing is stopped, as shown in Figure 4-9.



The screenshot shows the IBM CICS Explorer interface. The main window displays the 'Event Processing' tab for the context 'CNX0211I Context: EPRED7. Resource: EVNTGBL. 1 records collected at 09-Jul-2009 17:39:54'. Below this, a table shows the status of event processing for the EPRED7 region.

Region	Event Processing Status	Put Events	Current Event Capture Queue	Current Transactional Queue
EPRED7	STOPPED	0	0	0

Figure 4-9 Event processing stopped

4. To restart event processing, right-click to select the region for which you want to start event processing and click **Start**. Click **OK**.

4.4 Creating and installing a bundle definition

In this IBM Redbooks publication we use the CICS Explorer to define and install bundle resources other mechanisms such as CEDA can also be used.

Using the CICS Explorer we show how we perform the following tasks:

- ▶ Create a new bundle resource definition (BUNDDEF resource).
- ▶ Install a bundle resource in to CICS.

Bundles and event bindings

A CICS bundle resource is a new type of resource introduced in CICS TS V4.1. CICS bundles are used for a number of types of resource, one of which is the event binding. To enable an event binding we have to install a bundle resource into CICS.

When a bundle is installed into CICS, the various resources that it contains are each installed as a result. To install an event binding into CICS, install the bundle in which it is included.

When an event binding is installed, the capture specifications associated with the events in the bundle are deployed into CICS and the runtime will start to capture events where they match the filtering specified in the capture specifications. You

can deploy your event binding file to a CICS system by exporting a bundle directly to an z/FS file system. We use HFS to refer to the z/OS UNIX file system or zFS.

You enable an event binding by installing the resource object for the bundle to make it available to the CICS region. You can install a bundle using IBM CICS Explorer or the CICSplex Web user interface or by the RDO CEDA transaction or the DFHCSDUP utility.

Note: We show how we define and install a bundle, which we name “SHOPBUND” and which contains our sample event binding called “ShoppingEventBinding” as supplied in Appendix A, “Additional material” on page 347.

4.4.1 Creating a new bundle definition (BUNDDEF resource)

We define our bundle, which we call “SHOPBUND” and which contains our sample event binding file. It is located in z HFS directory, as shown in Example 4-1.

Example 4-1 Sample event binding

/u/cicsrs7/bundles/ShoppingBundle/	
Type	Filename
— Dir	.
— Dir	..
— Dir	META-INF
— File	ShoppingEventBinding.evbind

We use the New Wizard in CICS Explorer to create a BUNDDDEF resource definition that can then be installed in one or more active CICS systems. See Figure 4-10.

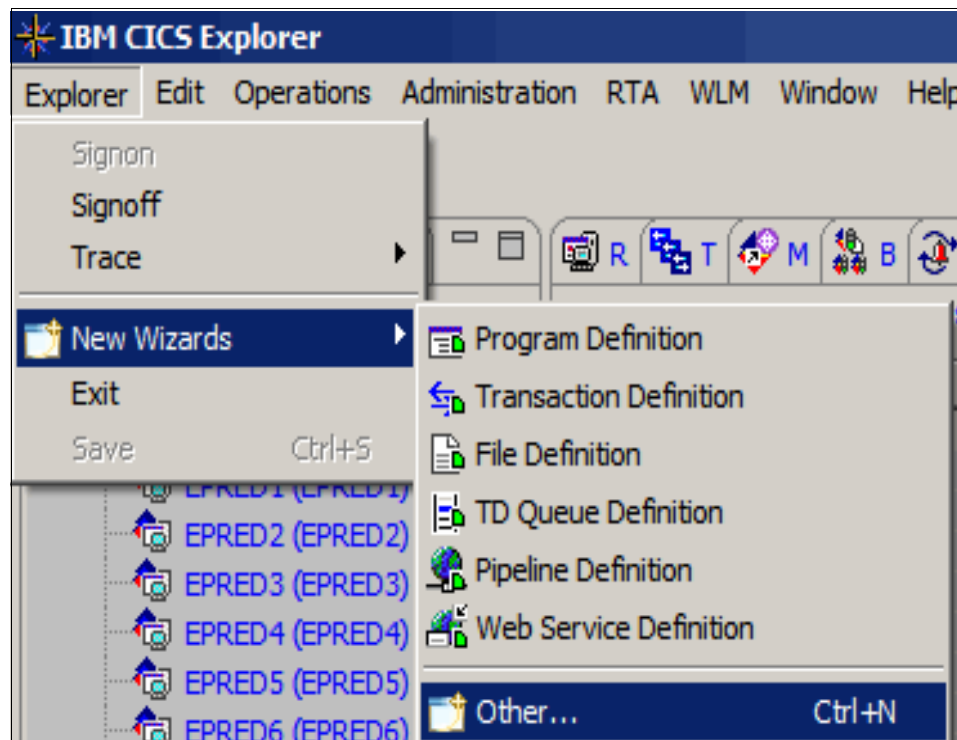


Figure 4-10 New Wizard in the CICS Explorer

We enter the basic details needed to create the resource using the New Wizard and add additional information using the editor. You must be connected to a CICS version V4 system to create resource definitions.

To create a new resource, perform the following steps:

1. Open the New Wizard using the following method:
 - a. Click **Explorer New Wizards**. Click **Other**.
 - b. Click **Next**.
 - c. Select **Bundle Definition** and click **Next**, as shown in Figure 4-11 on page 69.

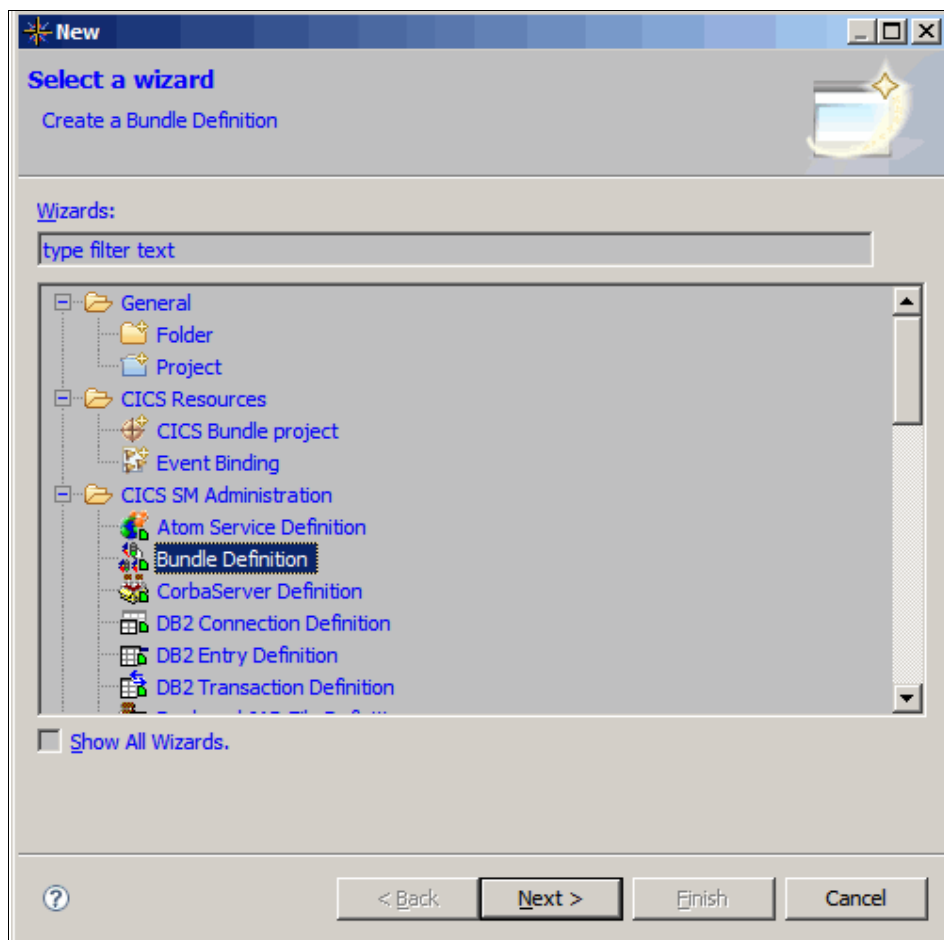


Figure 4-11 Creating a new BUNDEF

2. Click **Next** and the “New Bundle Definition” panel (Figure 4-12) displays.

New Bundle Definition

Create Bundle Definition

Data Repository: EPRED Browse...

Resource Group: Browse...

Name: SHOPBUND

Description: Shopping bundle definition

Bundle Directory: /u/cicsrs7/bundles/ShoppingBundle/ Browse...

☒ Open editor

< Back Next > Finish Cancel

Figure 4-12 Specifying bundle definition options

3. If you selected a CICSplex before opening the wizard, the Data Repository field contains the name of the CICSplex. You can overwrite the name or, if the field is empty, type a new name. The light-bulb symbol alongside the text field shows that content assist is available. When you press Ctrl +space, content assist displays a list of the possible choices for the field. You can double-click a name in the list to select the CICSplex.

4. If you selected a resource group before opening the wizard, the Resource Group field contains the name of the resource group, and the Data Repository field contains the name of the CICSplex. You can overwrite the name or, if the field is empty, type a new name. We leave the Resource Group field blank as we are not creating a resource group.
5. Complete the remaining fields in the wizard. We specify a Bundle Directory value of `/u/cicrs7/bundles/ShoppingBundle/`, as we have previously exported our sample event binding to that z HFS directory
6. If you want to save the new resource and immediately open it in the editor, ensure that the Open editor check box is selected.
7. Click **Finish** and your bundle definition is created. In our case, this means that our new definition is saved onto the CPSM repository.

Note: When creating a new definition with the CICS Explorer, the definition panel that opens for your specified new resource prompts you to supply the required parameters to complete the definition. Field help is available by pressing the PF1 key on any of the parameter fields. If you wish to define further options to your definition, select the open editor and click **Finish**. The editor opens and you can specify other attributes for your definition

4.4.2 Installing a bundle definition into CICS

An event binding is enabled by installing the bundle resource (provided that resource specifies “enabled” as its initial state, and the install action succeeds).

Perform the following steps to install the bundle using the IBM CICS Explorer.

1. From the IBM CICS Explorer toolbar, click **Administration Bundle Definitions** to view the list of BUNDDEF resource definitions. See Figure 4-13.

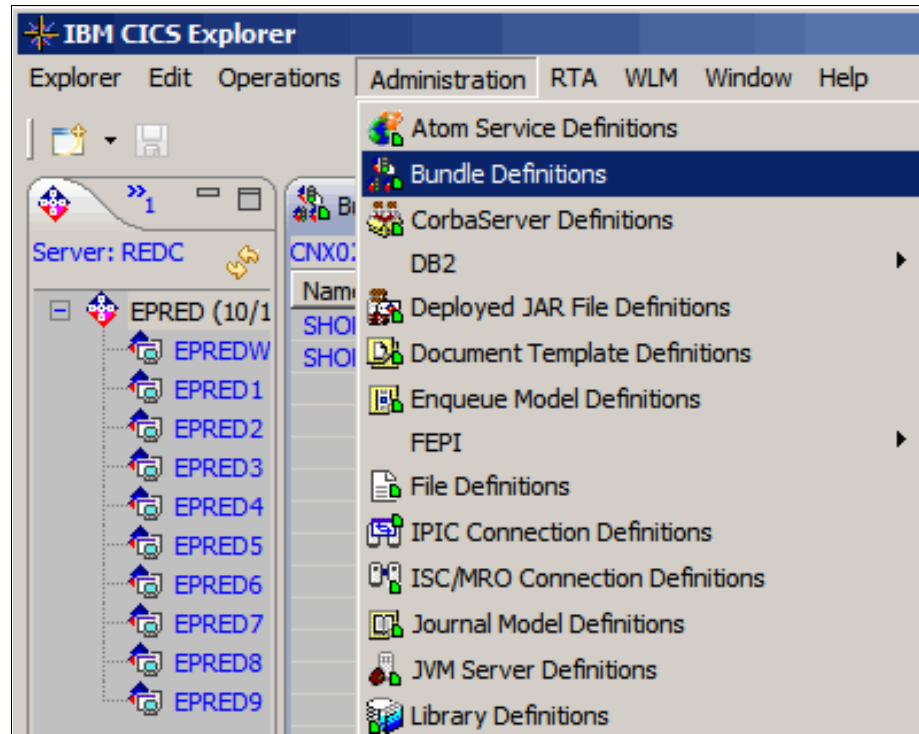


Figure 4-13 Bundle definitions to view the list of BUNDDEF resource definitions

2. Right-click the event binding bundle definition name in the Bundle Definitions view and click **Install**, as shown in Figure 4-14 on page 73. On the “Install action” panel, select the appropriate target CICS system or system group check box where you want to install your event binding bundle and click **OK**.

You get message, CNX0551I Install of BUNDDEF definitions into EPRED7 successful. The bundle containing the event binding installs in the specified CICS region. CICS also dynamically creates any other resources that are defined in the bundle.

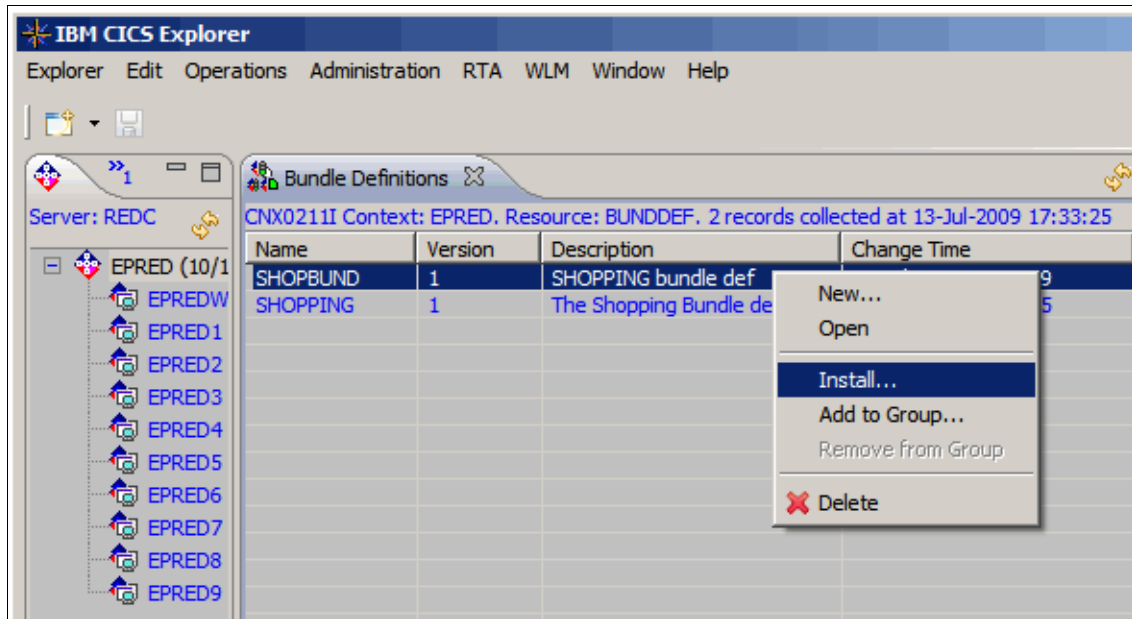


Figure 4-14 Install bundle definitions

What to do next

After the resource definition installs successfully, you can view the status of installed event bindings or bundles.

4.5 Enabling and disabling and discarding events

Using the IBM CICS Explorer, we show how to enable and disable and discard event bindings or bundles.

We show how we disable and discard an event binding in Figure 4-15 through Figure 4-25 on page 81. Select **Operations** Event bindings or Operations > Bundles. You can also disable and discard event bindings or bundles using a right-click in the Operations > Event bindings or Operations > Bundles views. See Figure 4-15.

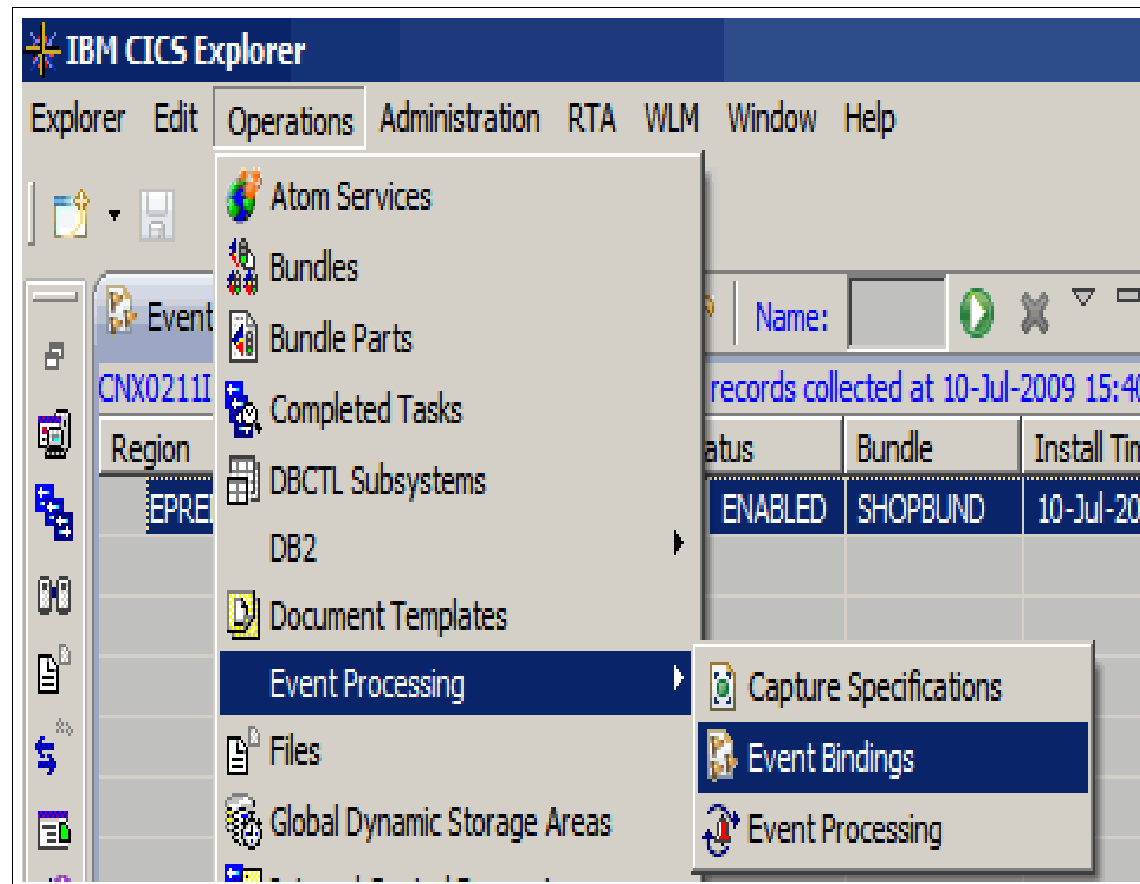


Figure 4-15 Working with event bindings

Click to select your event binding, then right-click to disable, as shown in Figure 4-16 on page 75.

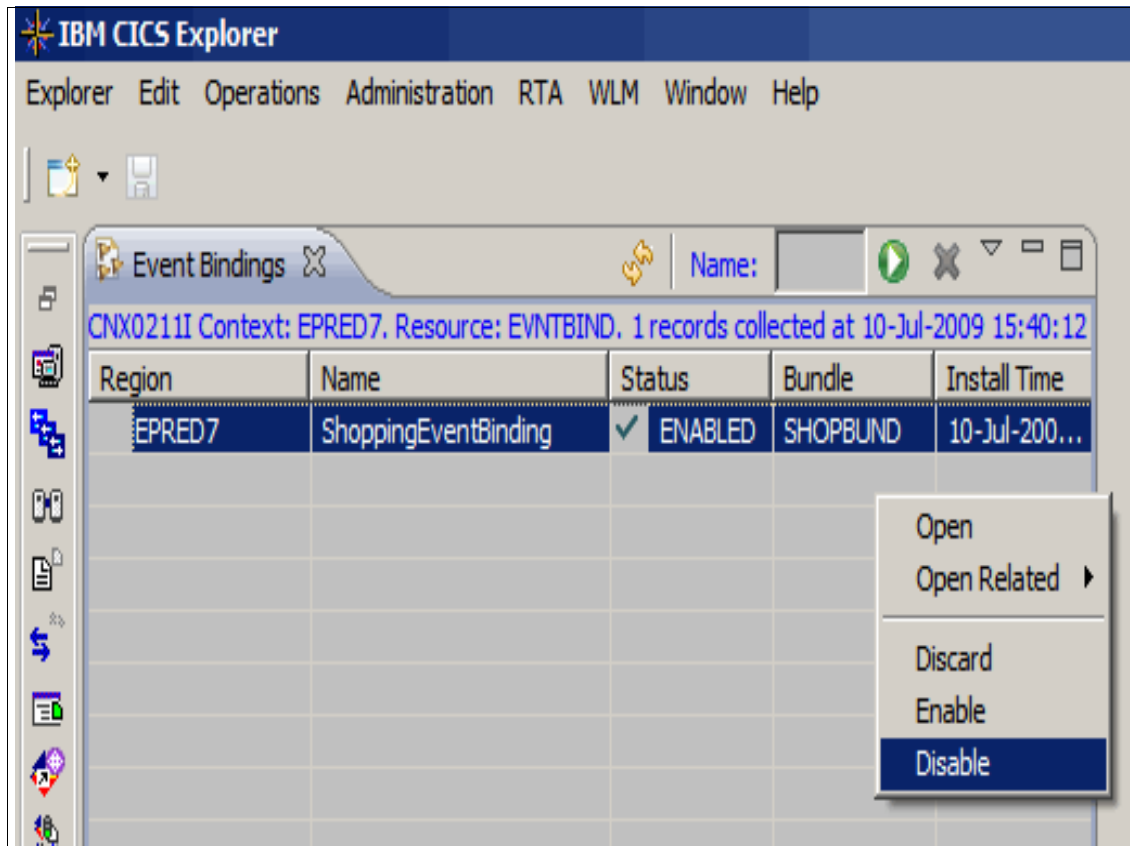


Figure 4-16 Disabling the 'ShoppingEventBinding' event binding

Click **OK**. See Figure 4-17. The event binding is disabled.

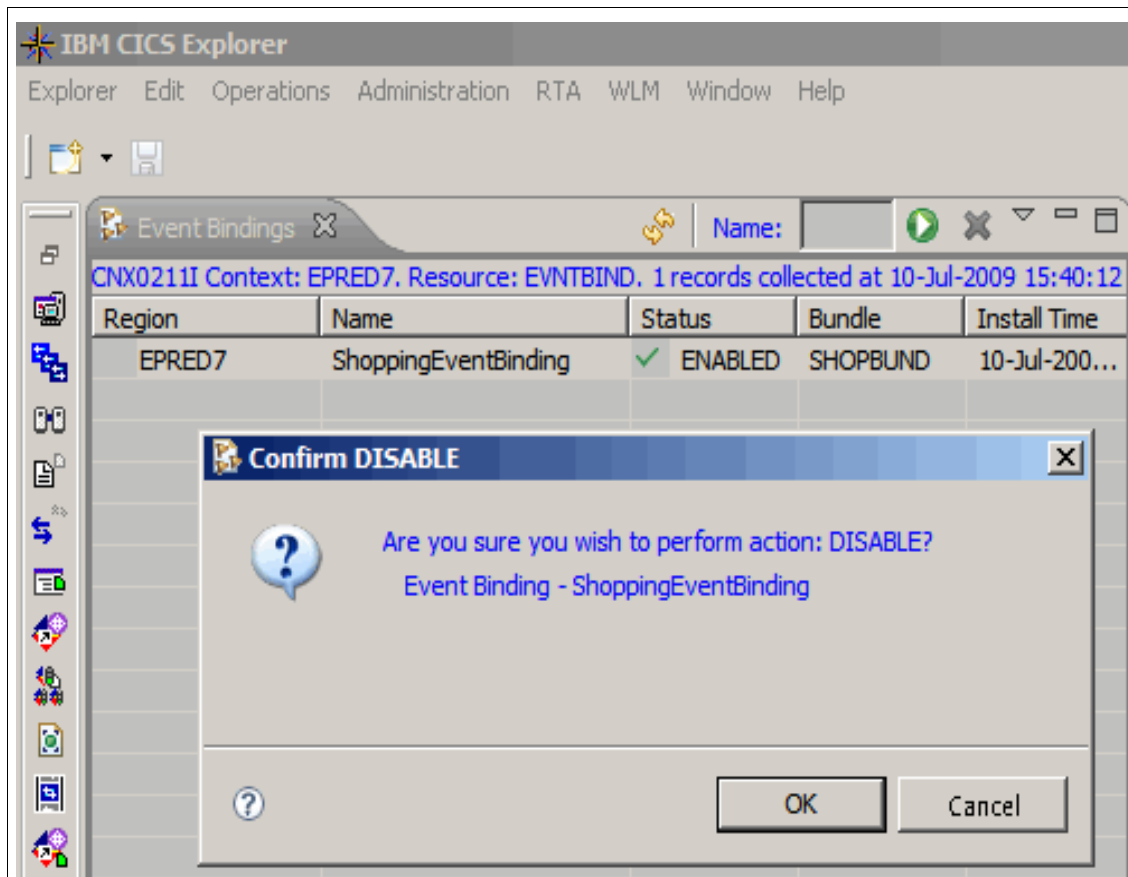


Figure 4-17 Confirmation panel for disable

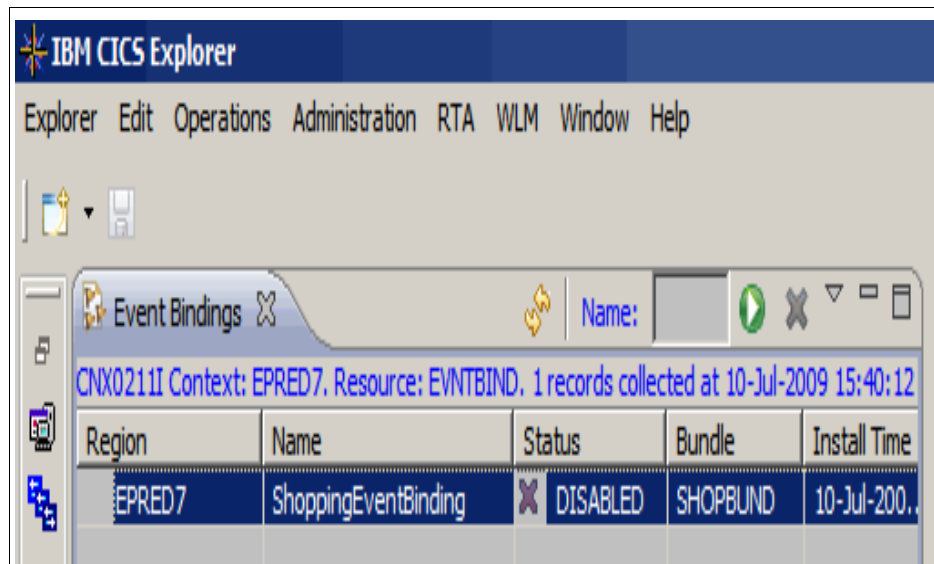


Figure 4-18 Disabled 'ShoppingEventBinding' event binding

Right-click and now we can discard the event binding as in Figure 4-19.

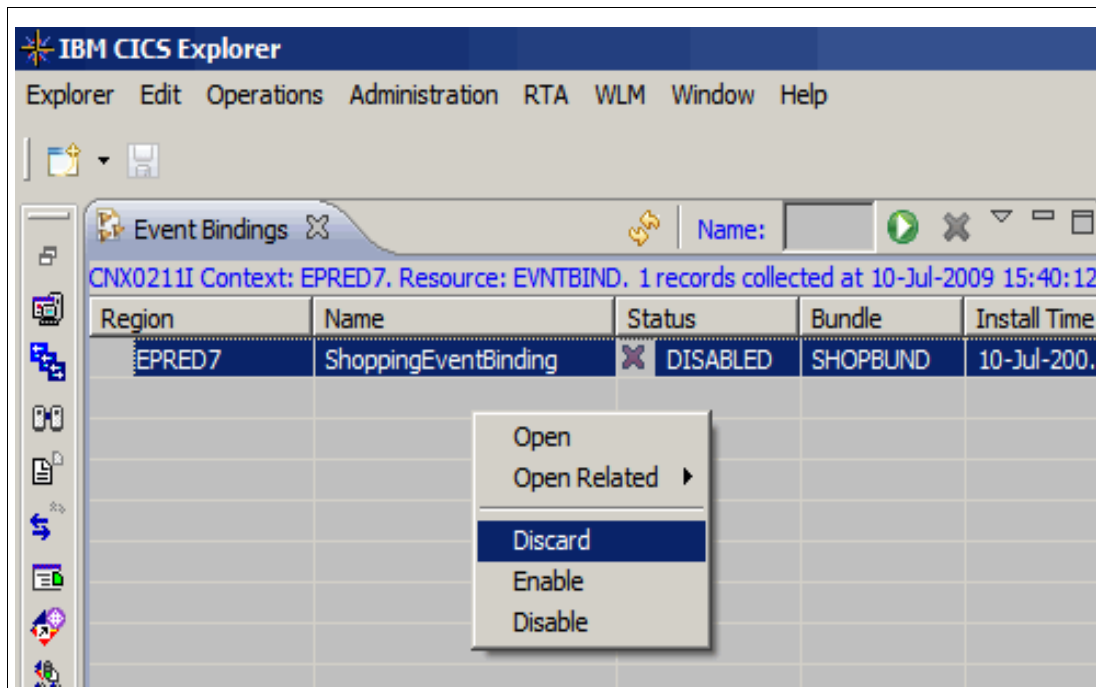


Figure 4-19 Discarding a disabled event binding

Click **OK** and the event binding is discarded. See Figure 4-20 on page 79.

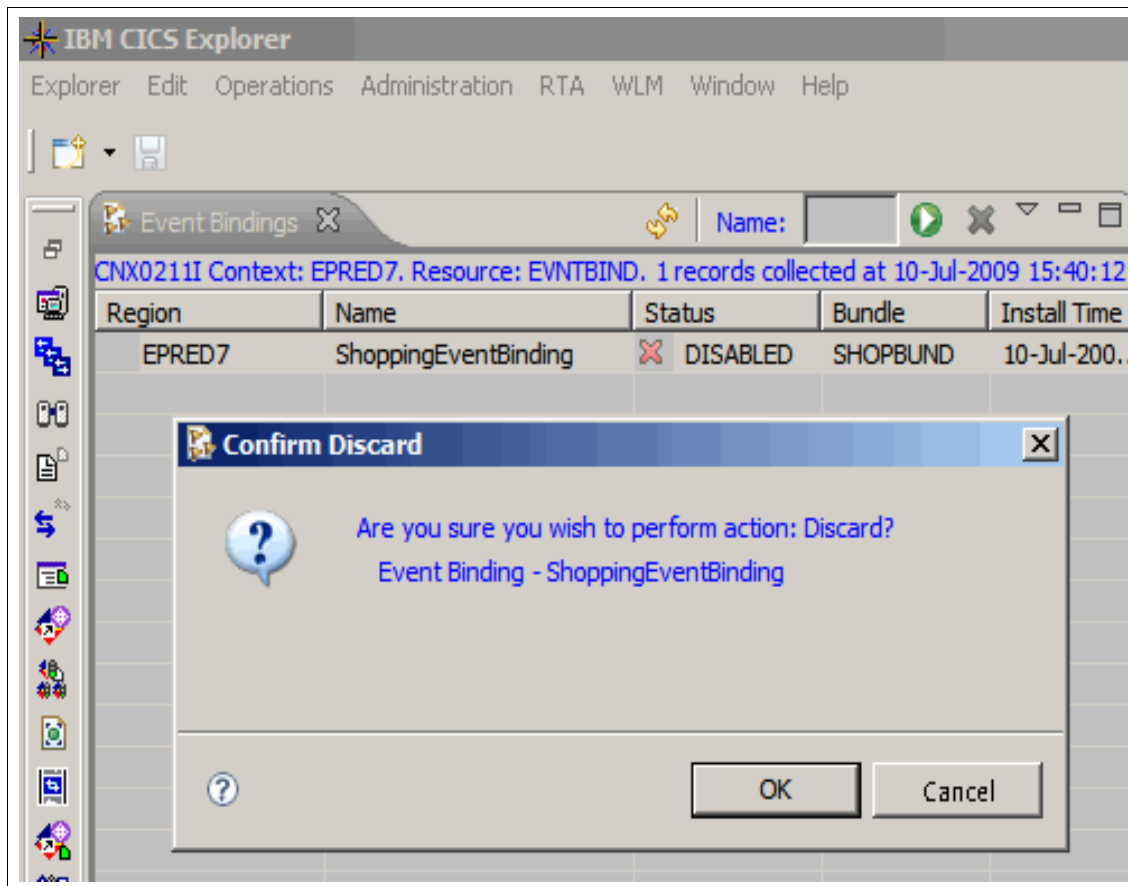


Figure 4-20 Confirm discard request

Note: When we disable an event binding the bundle state changes to disabled. If there are multiple event bindings within a bundle and we disable just one event binding, the other event bindings remain enabled even though the bundle will show as disabled. An attempt to discard a bundle in this state will fail, as shown in Figure 4-21 on page 80. If you disable a bundle explicitly, all parts it contains are disabled and the bundle can be discarded.

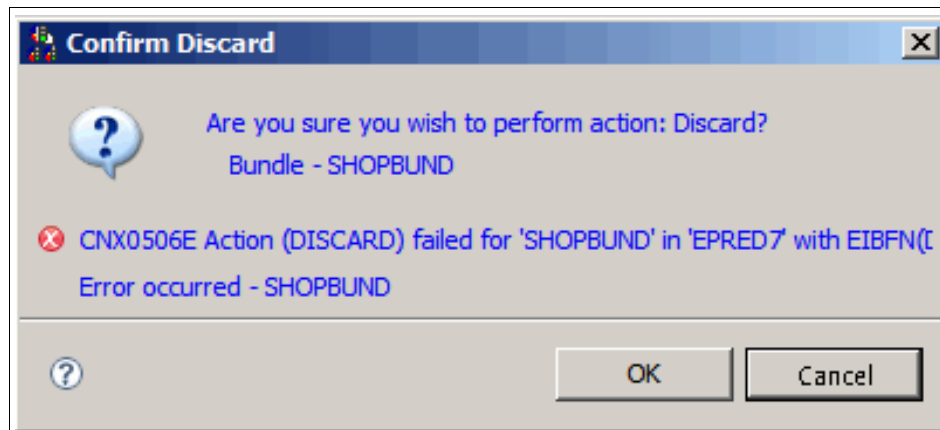


Figure 4-21 Discard of bundle fails -

4.5.1 Discarding a bundle

We show how to discard a bundle in the Figure 4-22, Figure 4-23 on page 81, and Figure 4-24 on page 81.

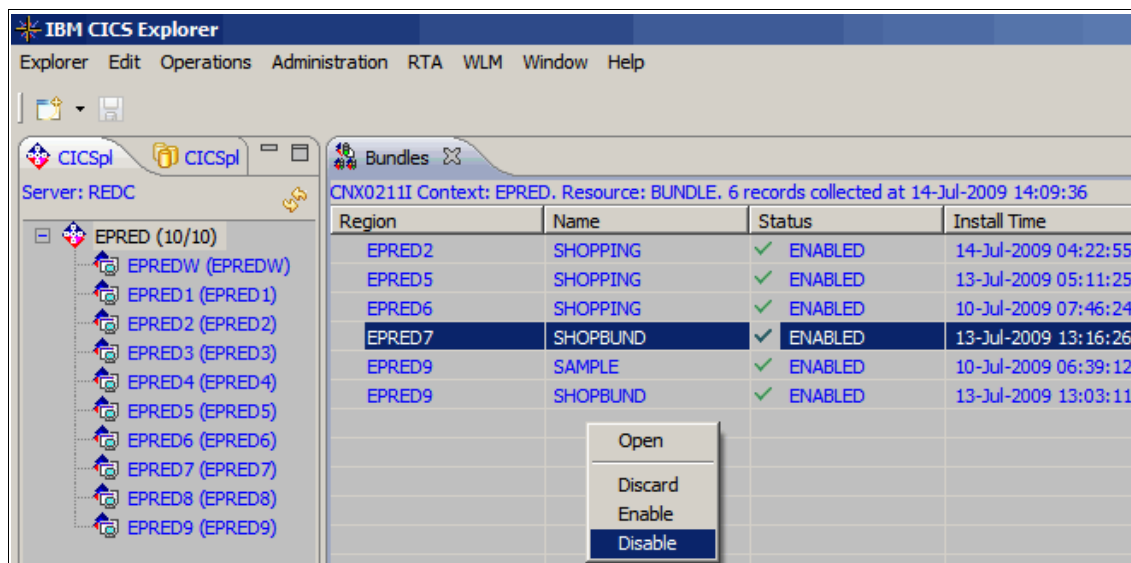


Figure 4-22 Disabling a bundle

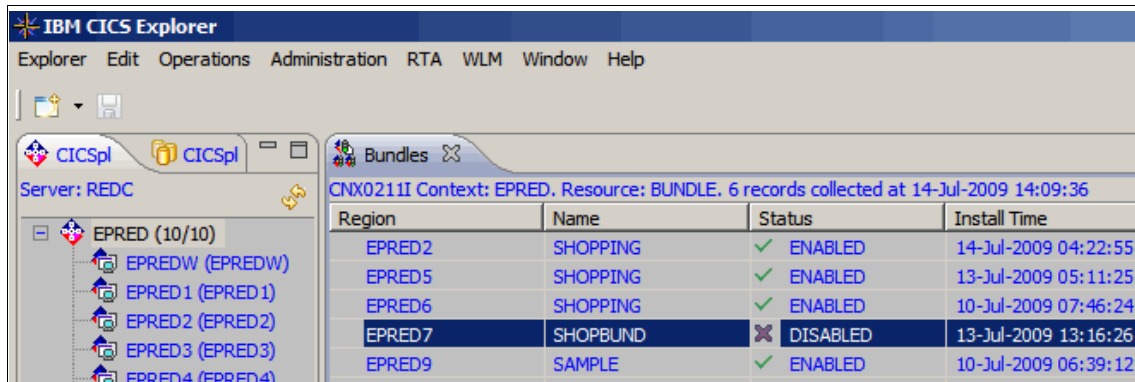


Figure 4-23 Bundle disabled

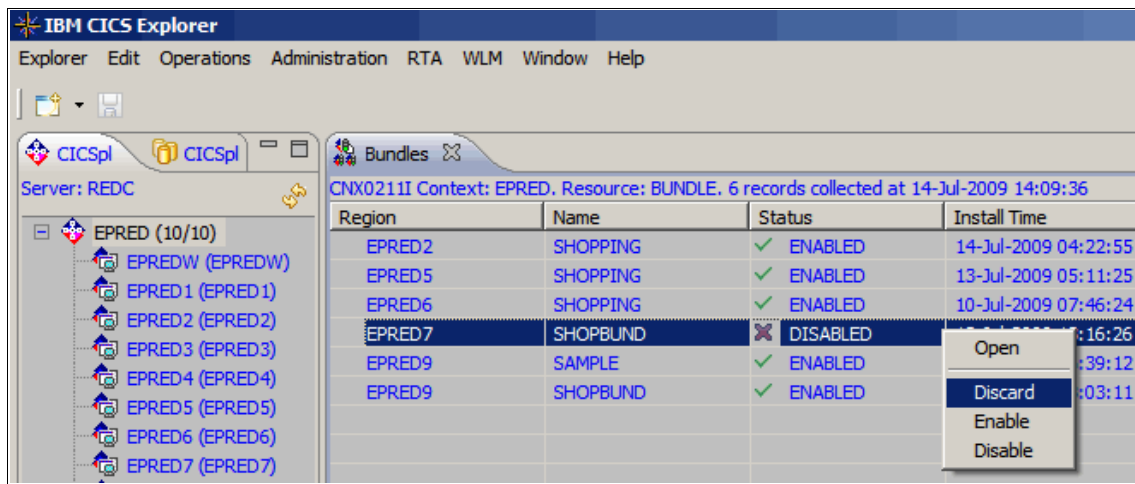


Figure 4-24 Discarding the bundle

After the bundle is discarded it no longer appears in our list of bundles and we get a discard successful message as in Figure 4-25.

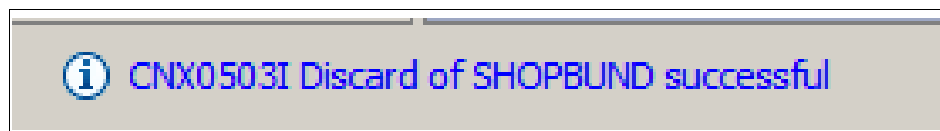


Figure 4-25 Bundle discarded message

4.5.2 Replacing a deployed bundle

If you want to replace a deployed bundle after it is installed, there are 2 ways to do this:

- ▶ Disable, discard, then install the changed version of a bundle with the same name. No events will be emitted from the moment the bundle is disabled until the moment the install completes successfully.
- ▶ You can replace a bundle without disabling the event binding by creating a new bundle, which must have a different name from the original bundle and contains the event binding with the same name. Events will continue to be emitted until the bundle install completes successfully, at which point the new binding will replace the previous version.

4.6 Security considerations for CICS events

To support event processing in CICS TS V4.1 some new security-related resources have been added. In this section we outline these new resources and show how we set up CICS and the external security manager (in our case RACF) to protect BUNDLES and EVENTBINDINGS from unauthorized access.

4.6.1 Changes to security

Resource and command security apply to BUNDLE, CAPTURESPEC, EVENTBINDING, and EVENTPROCESS resource when these functions are enabled for the CICS region.

New category 1 transactions

The CEPD transaction is a category 1 transaction that is implemented by program DFHEPDS.

The CEPM transaction handles the captured event queue. It distributes events to be formatted and emitted. The CEPM transaction is a category 1 transaction that is implemented by program DFHEPSY. Both CEPD and CEPM are defined internally by the event processing domain. The CRLR transaction is a category 1 transaction and is used for bundle resource resolution.

Resource security

Resource security for EVENTBINDING resources uses resource profiles in the RCICSRES class or the WCICSRES grouping class, or equivalent customer-defined classes specified in the XRES system initialization parameter.

You must supply a prefix of EVENTBINDING to the name of the EVENTBINDING resource definition. We show an example of this in Example 4-2 on page 84.

Command security

Command security for CAPTURESPEC resources uses the CAPTURESPEC resource in the CCICSCMD class or the VCICSCMD grouping class.

Command security for EVENTBINDING resources uses the EVENTBINDING resource in the CCICSCMD class or the VCICSCMD grouping class.

Command security for EVENTPROCESS resources uses the EVENTPROCESS resource in the CCICSCMD class or the VCICSCMD grouping class.

Security using the XRES resource security parameter

The XRES system initialization parameter is used to security check the following CICS resources: ATOMSERVICE, BUNDLE, DOCTEMPLATE, EVENTBINDING, JVMSERVER, and XMLTRANSFORM. The BUNDLE and EVENTBINDING resources are the ones of interest for event processing.

CICS profiles are passed to the security manager for checking. For further details refer to CICS infocenter through the following Web page:

https://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=/com.ibm.cics.ts.doc/dfht5/topics/dfht5_doct.html

4.6.2 Setting up CICS security for event bindings

In our CICS region, we show how we used an ESM (in our case RACF) to protect access the eventbinding resource.

Our CMAS has simulated CICS Resource Security enabled.

In the SIT for CICS EPRED7, we set the security options detailed in Figure 4-26, but XRES is the specific option needed to enable resource checking.

```
SEC=YES
XAPPC=NO
XCMD=NO
XDB2=NO
XDCT=NO
XEJB=NO
XFCT=NO
XJCT=NO
XPCT=NO
XPPT=NO
XPSB=NO
XRES=YES
XTRAN=NO
XTST=NO
```

Figure 4-26 CICS SIT security options

RACF definitions

We want protect our BUNDLE, which we named SHOPBUND. This bundle contains an EVENTBINDING called ShoppingEventBinding.

From the TSO command prompt, we issue the following commands to define profiles in RACF for our BUNDLE and EVENTBINDING in the RCICSRES class. We permit user ID =CICSR7 to these profiles as shown in Example 4-2,

Example 4-2 RACF Permissions for BUNDLE and EVENTBINDING

```
RDEFINE RCICSRES BUNDLE.SHOPBUND UACC(NONE)
PERMIT BUNDLE.SHOPBUND CLASS(RCICSRES) ID(CICSR7) ACCESS(ALTER)
RDEFINE RCICSRES EVENTBINDING.ShoppingEventBinding UACC(NONE)
PERMIT EVENTBINDING.ShoppingEventBinding CLASS(RCICSRES) ID(CICSR7)
ACCESS(ALTER)
```

Note: RACF profile definitions and security checking are case sensitive.

We issue RACF SETROPTS to refresh the RCICSRES class. We specify ALTER access above as this is needed to allow our user ID “CICSR7” to DISCARD an event binding or bundle

With these options set, using the CICS Explorer, CICS user ID “CICSRS7” is able to see bundle SHOPBUN and event binding ShoppingEventBinding. This user ID is also able to INSTALL a bundle definition(BUNDEF) and DISABLE or DISCARD this event binding and bundle.

A user ID other than “CICSRS7” is unable to see the eventbindings or bundles which are installed in our CICS region. An attempt to access these resources does not generate a security error messages in the CICS Explorer. This is a feature of how CICS works. To help explain, we compare the behavior for an unauthorized user of the CICS Explorer with the way in which CEMT INQUIRE EVENTBINDING works.

For example, using CEMT INQUIRE EVENTBINDING(*) with generic target will generate a NOTFOUND response, even when the user ID issuing the command is not authorized to the EVENTBINDING profile, as we shown in Figure 4-27.

```
I EVENTBINDING(*)
STATUS: RESULTS - OVERTYPE TO MODIFY
Eventb(*)                                NOT FOUND

                                                                    SYSID=RED7
APPLID=EPRED7
RESPONSE: 1 ERROR TIME: 06.30.15  DATE: 07/14/09
```

Figure 4-27 CEMT INQUIRE EVENTBINDING generic

Whereas, a CEMT INQUIRE EVENTBINDING, which specifies the event binding name will get a NOT AUTHORIZED response, as we shown in Figure 4-28.

```

I EVENTB(SHOPPINGEVENTBINDING)
STATUS: RESULTS - OVERTYPE TO MODIFY
Eventb(SHOPPINGEVENTBINDING )           NOT AUTHORIZED

                                           SYSID=RED7
APPLID=EPRED7
  RESPONSE: 1 ERROR                       TIME: 09.35.53
DATE: 07/10/09
PF 1 HELP      3 END      5 VAR      7 SBH 8 SFH 9 MSG 10 SB 11
SF

```

Figure 4-28 CEMT INQ EVENTBINDING specific

In the CICS job log we see the security violation message which shows that the CICS default user “CICSUSER” is not authorized, as shown in Figure 4-29.

```

DFHXS1111 07/14/2009 09:35:53 EPRED7 CEMT Security violation by user
CICSUSER for resource EVENTBINDING.SHOPPINGEVENTBINDING in class
RCICSRES. SAF codes are (X'00000008',X'00000000'). ESM codes are
(X'00000008',X'00000000').

```

Figure 4-29 CICS job log showing security violation

Because the CICS Explorer obtains all of the attributes for all event bindings and then applies a filter to the results, it does not generate a security violation.

For example, if a user signed on to the CICS Explorer attempts to see event bindings for which they are not authorized, the event bindings are not displayed.

Details of the CICS Explorer signed-on user ID and password are entered in the CICS Explorer connection panel as shown in Figure 4-2 on page 60.

Note: To install a bundle resource (BUNDDEF) from the CICS Explorer connected to CICSplex SM Web User Interface and using CICSplex simulated resource security (as in our case), a CICSplex APAR PK91529 is required.

4.6.3 The user ID in event bindings

When you install a BUNDLE resource that includes an event binding for which you specified a user ID in the Adapter tab of the CICS event binding editor, CICS checks that the user ID performing the install operation is authorized as a surrogate user of the user ID specified in the CICS event binding editor. This check also applies to the CICS region user ID during group list install on a CICS cold or initial start.

We specify a user ID “CICSR9” in the Adapter, as shown in Figure 4-30. Then, we show how to set up the security authorization required to install the bundle which contains this event binding during an initial start of CICS with an RDO definition added to the CICS group list.

Adapter

Choose the adapter to emit events produced by this binding.

Adapter **TS Queue**

The TS Queue EP adapter emits events to a named CICS TS queue. This adapter can be used to validate that the correct events are being captured with the correct data. This adapter can be used to emit events to any consumer which reads from a TS queue.

Queue Name **QueryEventTSq**

System ID (Optional)

Use Auxiliary Temporary Storage ☐

Export Event Specifications...

Advanced Options

These optional dispatcher settings are for advanced users.

Dispatch Priority **Normal**

Transaction ID

User ID **CICSR9** ☐ Use Context User Id

System ID

Events are Transactional ☐

Figure 4-30 Adapter with USER ID specified

With user ID “CICSR9” set in the EP adapter, we attempt to INITIAL start CICS and install the bundle SHOPBUN through an RDO group (PHILBUND) included in the CICS grouplist. The of install of our event binding SimpleShoppingEventbinding now fails with a security violation as expected, as shown in Example 4-3 on page 89.

Example 4-3 Security violation

```
DFHRL0107 I 07/15/2009 12:48:38 EPRED7 CICSRS1 The CICS resource life-cycle manager
has started to create the BUNDLE resource SHOPBUND.
DFHXS1111 07/15/2009 12:48:38 EPRED7 CSSY Security violation by user CICSRS1 for
resource CICSRS9.DFHINSTL in class SURROGAT. SAF codes are (X'00000004',X'00000000').
ESM codes are (X'00000004',X'00000000').
DFHEC1010 07/15/2009 12:48:38 EPRED7 Userid CICSRS1 is not authorized to create
EVENTBINDING ShoppingEventBinding with an EP adapter userid of CICSRS9.
DFHRL0102 E 07/15/2009 12:48:38 EPRED7 CSSY The CICS resource life-cycle manager
failed to create the resource ShoppingEventBinding
and returned with reason CALL_BACK_ERROR.
DFHAM4893 I 07/15/2009 12:48:38 EPRED7 Install for group PHILBUND has completed
successfully.
```

We issue the following RACF commands in TSO to allow the install to work, as shown in Example 4-4.

Example 4-4 RACF command

```
RDEFINE SURROGAT CICSRS9.DFHINSTL UACC(NONE) OWNER(CICSRS9)
PERMIT CICSRS9.DFHINSTL CLASS(SURROGAT) ID(CICSRS1) ACCESS(READ)
```

We issue RACF SETROPTS to refresh the SURROGAT class profile, and then we try to INITIAL start CICS. This time it is successful, as shown in Example 4-5.

Example 4-5 Initial start of CICS

```
DFHRL0107 I 07/15/2009 12:58:14 EPRED7 CICSRS1 The CICS resource life-cycle manager
has started to create the BUNDLE resource
SHOPBUND.
DFHEC1001 07/15/2009 12:58:14 EPRED7 Event binding ShoppingEventBinding installed
successfully.
DFHRL0109 I 07/15/2009 12:58:14 EPRED7 CSSY The CICS resource life-cycle manager has
created the BUNDLE resource SHOPBUND and the
BUNDLE is in the enabled state.
DFHAM4893 I 07/15/2009 12:58:14 EPRED7 Install for group PHILBUND has completed
successfully.
```



Generating events

In this chapter we discuss the following procedures:

- ▶ Creating a BUNDLE project
- ▶ Creating an Event Binding
- ▶ Specifying adapters
- ▶ Installing a bundle definition
- ▶ Testing an event specification

5.1 Creating the HFS directories

Use the TSO ISHELL to create a HFS directory (Example 5-1), to which we later export the event binding file.

Example 5-1 Created HFS directory

`/u/cicsrs2/bundles`

5.2 Creating the bundle project

We use IBM CICS Explorer to create the event binding file.

1. Open IBM CICS Explorer, and select the resource perspective. Right-click in the Project Explorer view, click **New**, and click **CICS Bundle project**, as shown in Figure 5-1.

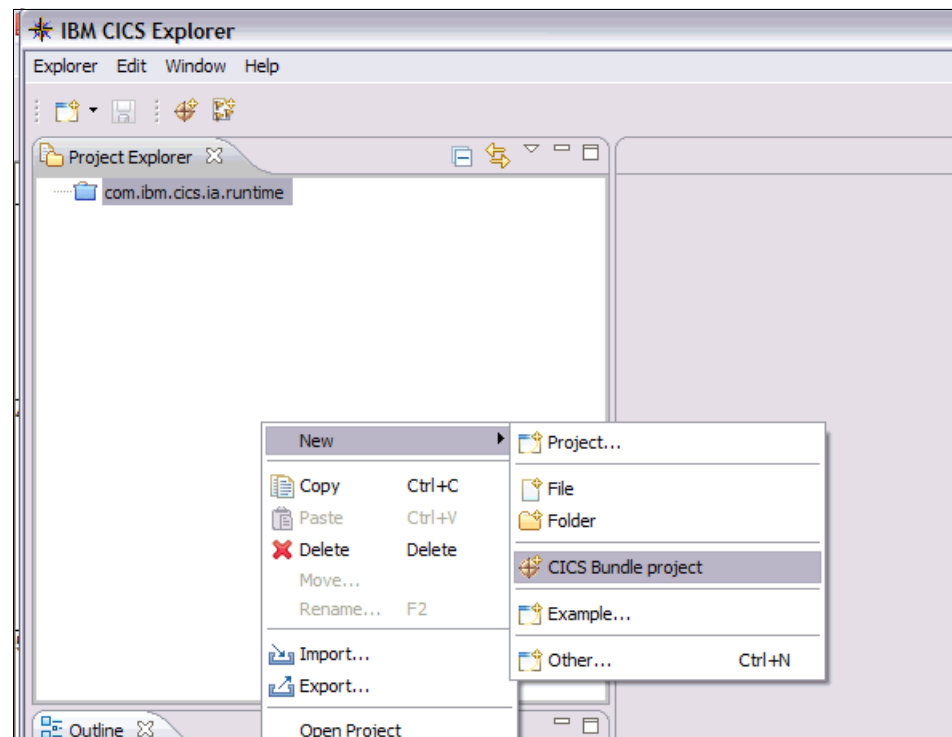


Figure 5-1 Creating CICS bundle project

The Bundle Project view displays.

2. Specify the project name, `ShoppingEventBundle`, as shown in Figure 5-2, and click **Finish**.

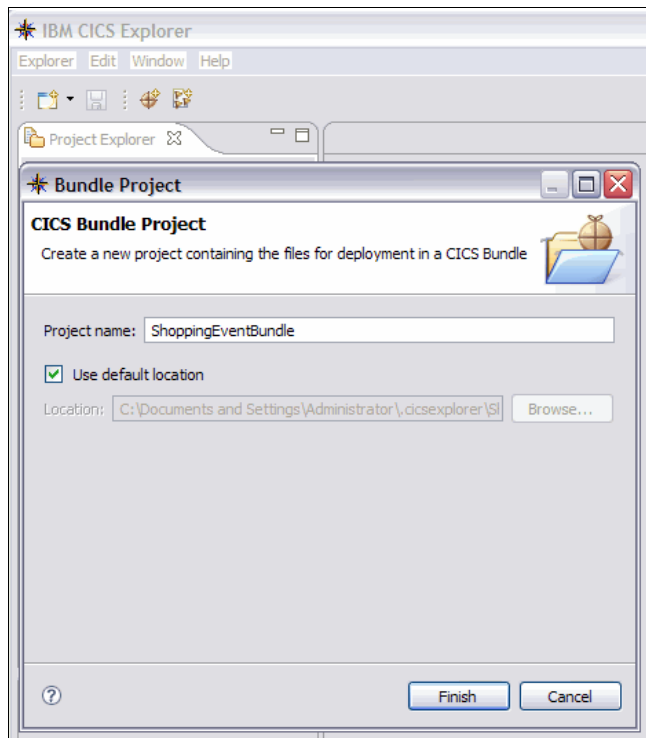


Figure 5-2 *ShoppingEventBundle*

3. Create the event binding within this bundle. In the Project Explorer view right-click the just created project, ShoppingEventBundle. Click **New Event Binding** as shown in Figure 5-3.

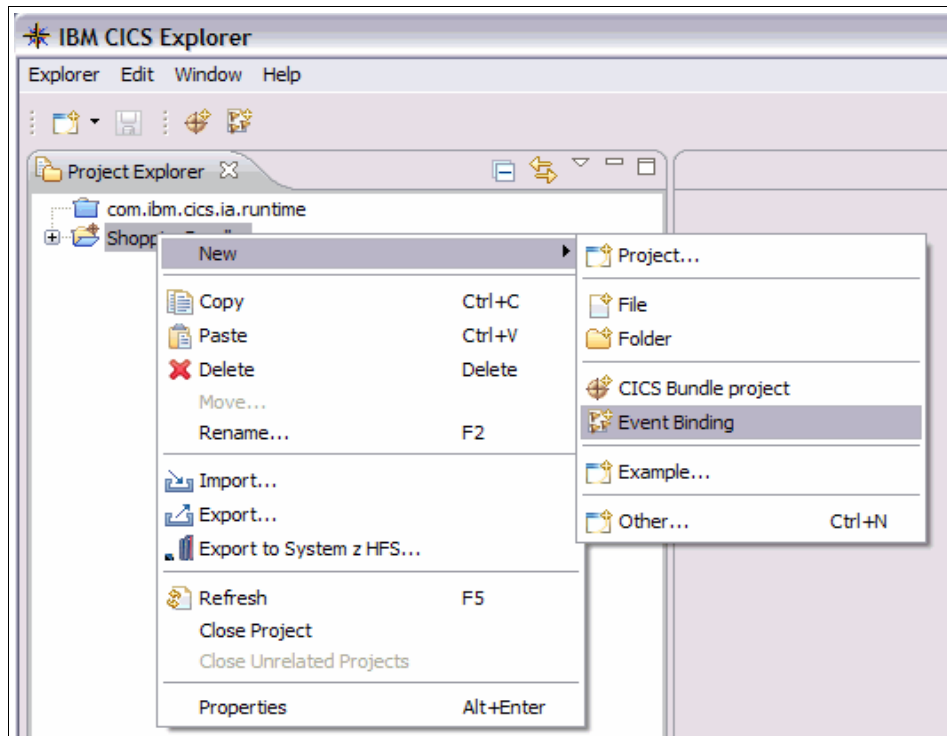


Figure 5-3 Creating event binding

4. Enter **ShoppingEventBinding** in the file name field and click **Finish**, as shown in Figure 5-4 on page 95.

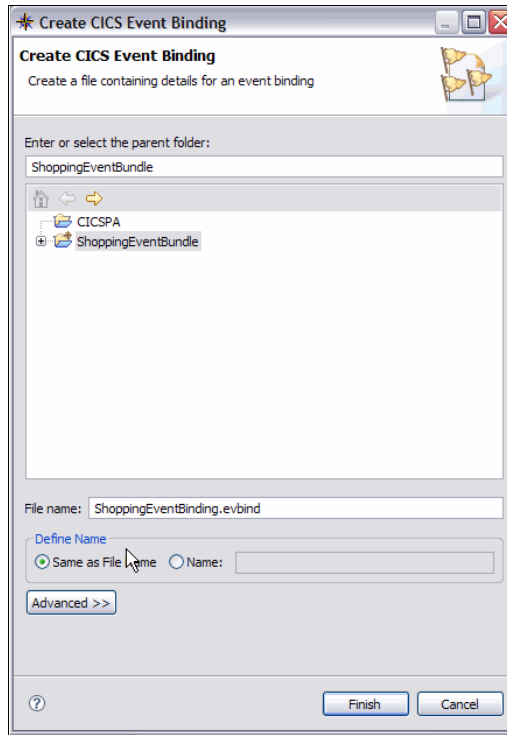


Figure 5-4 ShoppingEventBinding

Note: Figure 5-5 shows that some errors appear in the Problems pane, these are not concerns but are caused by information not yet entered.

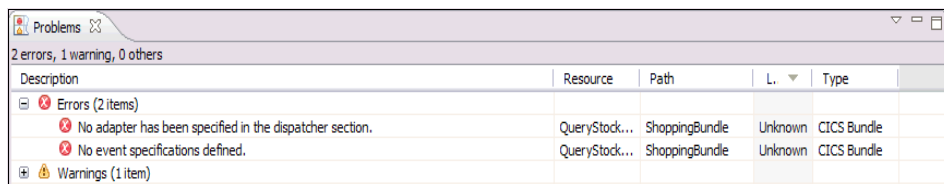


Figure 5-5 Creating event binding: Problems shown

5. Add an event specification to the event binding. In the view presented (Figure 5-6), enter a description for the event binding, and click **Add**.

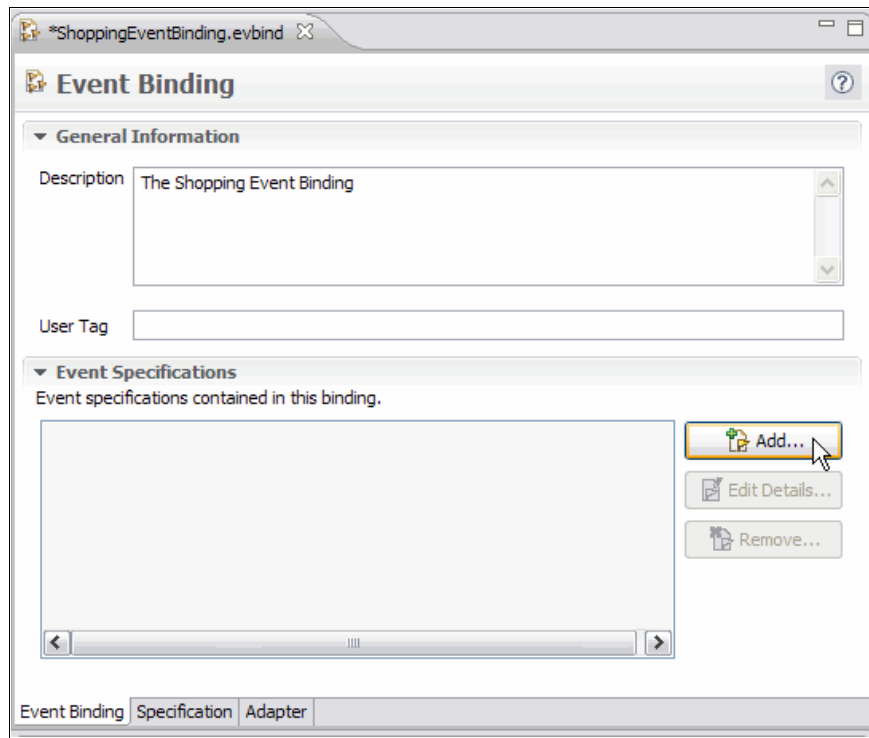


Figure 5-6 Add QueryStock binding specification

6. Enter the name of the Event Specification, QueryStock, a description, and click **OK** as shown in Figure 5-7 on page 97.

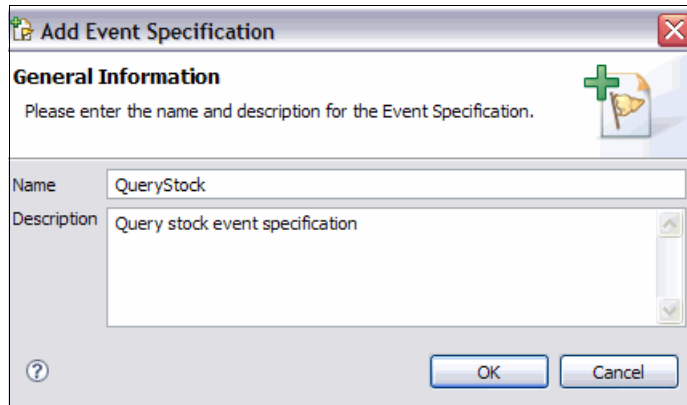


Figure 5-7 *QueryStock event specification*

We want the event to include the customer number and stock identification so we add these as items of emitted business information.

7. Click the **Specification** tab, and click **Add** next to the Emitted Business Information table. In the Emitted Business Information view, shown in Figure 5-8, enter the following values:
- Name: CustomerNumber
 - Type: Numeric
 - Length: 6
 - Precision: 0
 - Description: The customer number
- Click **OK**.

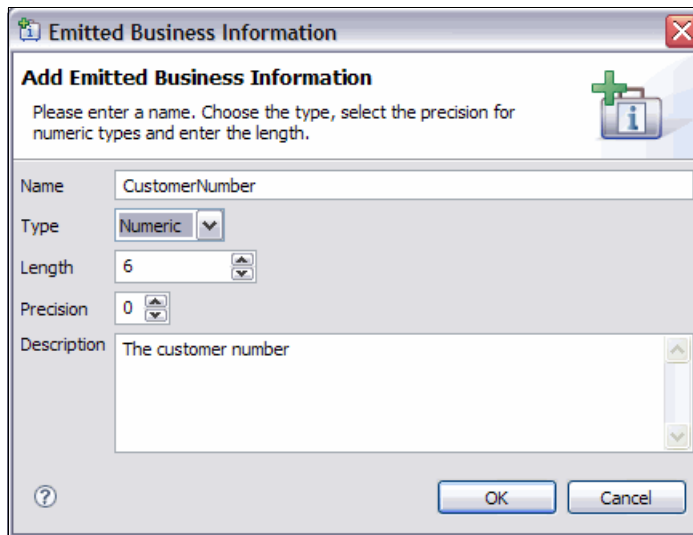


Figure 5-8 Add CustomerNumber business information

8. Repeat the previous task to add StockId business information, and enter the information, as shown in Figure 5-9 on page 99:
- Name: StockId
 - Type: Numeric
 - Length: 6
 - Precision: 0
 - Description: The stock identification
- Click **OK**.

The screenshot shows a Windows-style dialog box titled "Emitted Business Information" with a close button (X) in the top right corner. Inside the dialog, there is a section titled "Add Emitted Business Information" with a green plus icon and a brief instruction: "Please enter a name. Choose the type, select the precision for numeric types and enter the length." Below this, there are five input fields: "Name" with the text "StockId", "Type" with a dropdown menu showing "Numeric", "Length" with a numeric input of "6" and up/down arrows, "Precision" with a numeric input of "0" and up/down arrows, and "Description" with a text area containing "The stock identification" and a vertical scrollbar. At the bottom left is a help icon (question mark), and at the bottom right are "OK" and "Cancel" buttons.

Figure 5-9 Add StockId business information

9. Add a Capture Specification to indicate how CICS can capture this event from the application. In the Specifications view (Figure 5-10) click **Add a Capture Specification**.

Specifications

General
Identify and describe the event.

Name: QueryStock Edit...

Description: Query stock event specification

Emitted Business Information
Describe and order the business information to be emitted by the event.

Name	Type	Length	Precision	Description
CustomerNumber	Numeric	6	0	The customer number
StockId	Numeric	6	0	The stock identification

Add...
Edit...
Remove...
Move Up
Move Down

Capture Specifications
Add Capture Specifications to this event.

Add a Capture Specification...

Automatic Capture Specification
Use this to automatically generate a capture specification for a signal event call using the business information entered above.

Add an Automatic Capture Specification...

Event Binding | Specification | Adapter

Figure 5-10 Event Specification view

10. In the Add Capture Specification view, enter the information shown in Figure 5-11 on page 101:

- Name: CaptureQueryEvent
- Description: Capture the Query Event

Click **OK**.

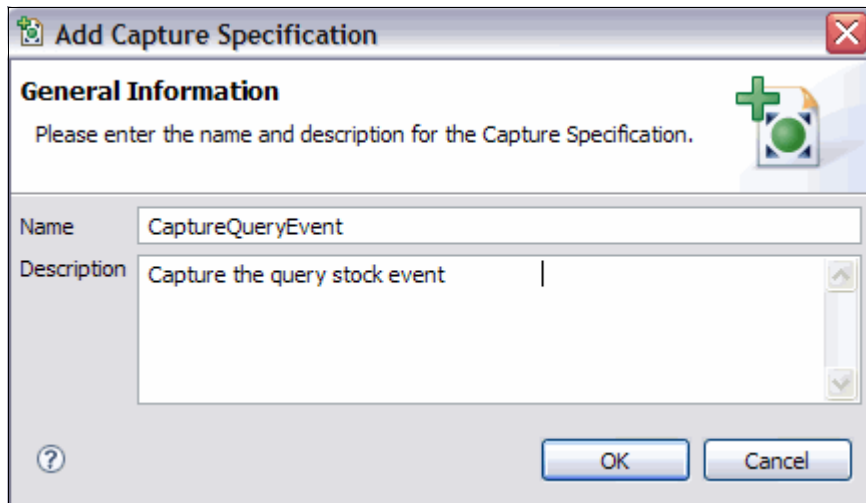


Figure 5-11 Add CaptureQueryEvent Capture Specification

11. In the next view, set the Capture Point to PROGRAM INITIATION and click **Next:Filtering**, as shown in Figure 5-12

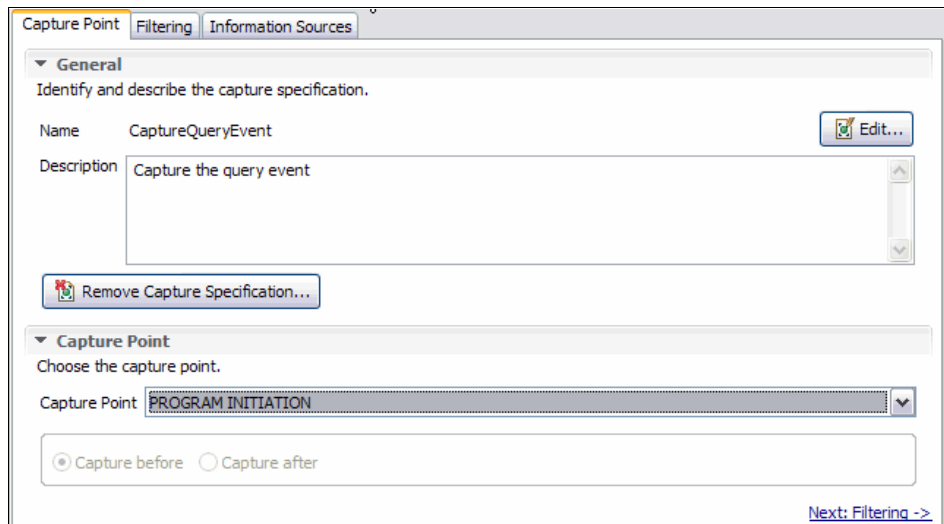


Figure 5-12 Specify capture point for Query Event

12. Set the filtering for the event. Only events that matches the following criteria will be emitted.

- Name: PROGRAM*
- Operator: Equals
- Value: QUERY

For this to happen we enter the Filtering information as shown in Figure 5-13. In the Application Context part we let all values default. In the Application Command Option we set the program operator to **Equals** and the value to **QUERY**, the name of our program, and let CHANNEL default to **All**. Click **Add** on the right side of the Application Data Table to add filtering based on values in the data.

The screenshot shows the 'ShoppingEventBinding.evbind' application context configuration window. The 'Filtering' tab is selected. The 'Application Context' section has four rows: 'Transaction ID', 'Current Program', 'User ID', and 'Response Code', all with 'All' in the 'Operator' dropdown and empty 'Value' fields. The 'Application Command Options' section has two rows: 'PROGRAM*' with 'Equals' in the 'Operator' dropdown and 'QUERY' in the 'Value' field, and 'CHANNEL' with 'All' in the 'Operator' dropdown and an empty 'Value' field. The 'Application Data' section has a table with columns 'Source', 'Container', 'Offset', 'Length', and 'Offset'. The table is empty. To the right of the table are buttons 'Add...', 'Edit...', and 'Remove...'. At the bottom are links '<- Back: Capture Point' and 'Next: Information Sources ->'. The bottom status bar shows 'Event Binding', 'Specification', and 'Adapter' tabs.

Figure 5-13 Specifying the application context

If the DISP-STOCK-ITEM container exists in the channel passed to the QUERY program, we know stock is being queried by the user.

13. In the Application Data Predicate view, in the Predicate section, set the Operator to **Exists**. In the variable location and format section, set Location to **CHANNEL**, and Container to the name of container used in our QUERY program, DISP-STOCK-ITE,M and click **OK**. See Figure 5-14 on page 103.

Application Data Predicate

Add New Application Data
Define the predicate, choose the location where the data is found then enter the format.

Predicate
Define Predicate
Operator:
Value:

Variable location and format
Location:
Container:
Enter the format or choose an item from an imported language structure.

Type:
Offset:
Length:
Precision:
Codepage:

Figure 5-14 Query stock Application Data Predicate

Tip: We only have one filter predicate in our example. If however more filter predicates are required, putting those that filter out the largest number of unwanted events before the filter predicates that exclude fewer unwanted events, in the event specification, gives the best performance. Unfortunately the only way to achieve this, is to add them in the right order. No reordering is possible.

14. Click **Next: Information Sources** on the Filtering panel (Figure 5-15) to indicate how CICS can capture the data requested as emitted business information.

ShoppingEventBinding.evbind

Specifications

QueryStock
CaptureQueryEvent

Capture Point **Filtering** Information Sources

▼ **Application Context**
Define predicates to filter events.

Context	Operator	Value
Transaction ID	All	
Current Program	All	
User ID	All	
Response Code	All	Ok

▼ **Application Command Options**
Define predicates for command options. Predicates marked with * should be specified to maintain CICS performance.

Name	Operator	Value
PROGRAM*	Equals	QUERY
CHANNEL	All	

▼ **Application Data**
Define predicates for application data. Whilst defining a predicate you may import a language structure and choose an item from it.

Source	Container	Offset	Length	Operator	V
CHANNEL	DISP-STO...	0	1	Exists	

Add... Edit... Remove...

<- Back: Capture Point Next: Information Sources ->

Event Binding | Specification | Adapter

Figure 5-15 Business information filtering

15. Because we already have added Emitted Business Information (Figure 5-8 on page 98) the “Information Source” panel (Figure 5-16 on page 105) has already been filled in. To fill in the information source, select the **CustomerNumber** business information and click **Edit**.

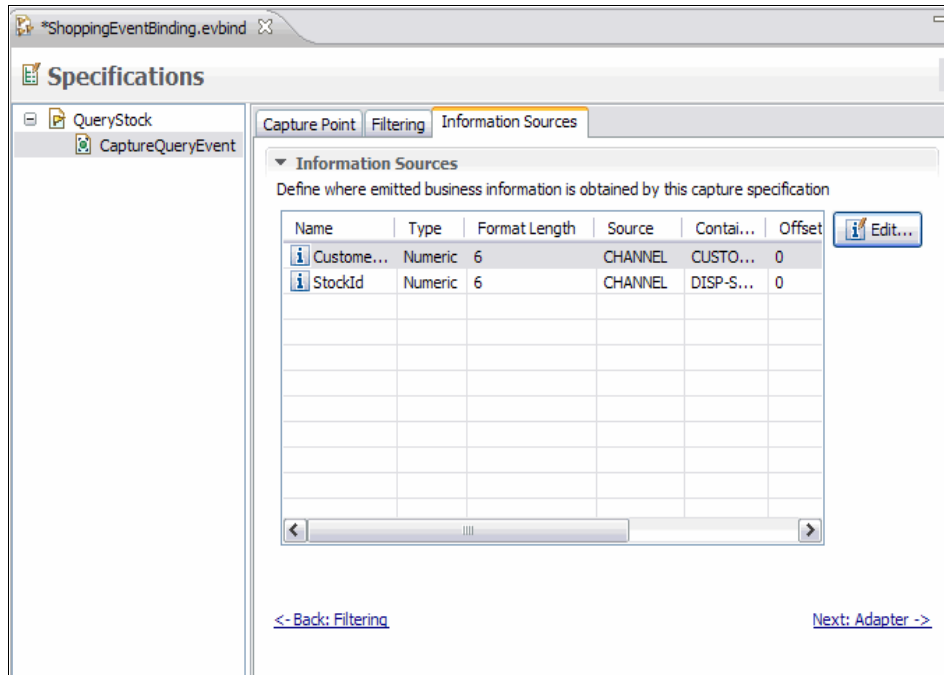


Figure 5-16 Information source panel

16. On the “Edit Information Source” panel in the Available Data section, click **CHANNEL** and enter the following values, as shown in Figure 5-17 on page 106.

Note: if you have the copybook describing the data in the container, then you can use that rather than having to know the offsets of data yourself.

- Application Data: CHANNEL
- Container: CUSTOMER-NO
- Type: Packed Decimal
- Offset: 0
- Length: 3
- Precision: 0

Note: The emitted business information source is packed decimal. Therefore, it has a length of three, whereas the emitted business information is string format and has a length of six.

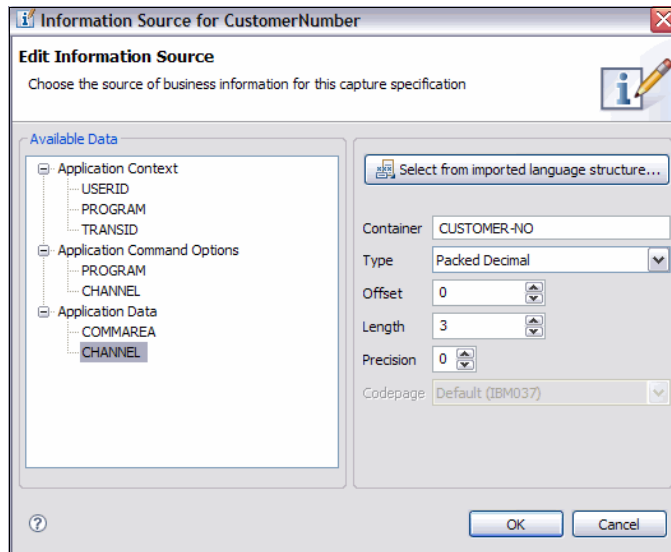


Figure 5-17 CustomerNumber information source

17. Repeat the previous task for the StockId business information with the following values.

- Application: DataCHANNEL
- Container: DISP-STOCK-ITEM
- Type: Packed Decimal
- Offset: 0
- Length: 3
- Precision: 0

Click **OK**. Figure 5-18 on page 107 shows where the emitted business information will be obtained.

[illegible]

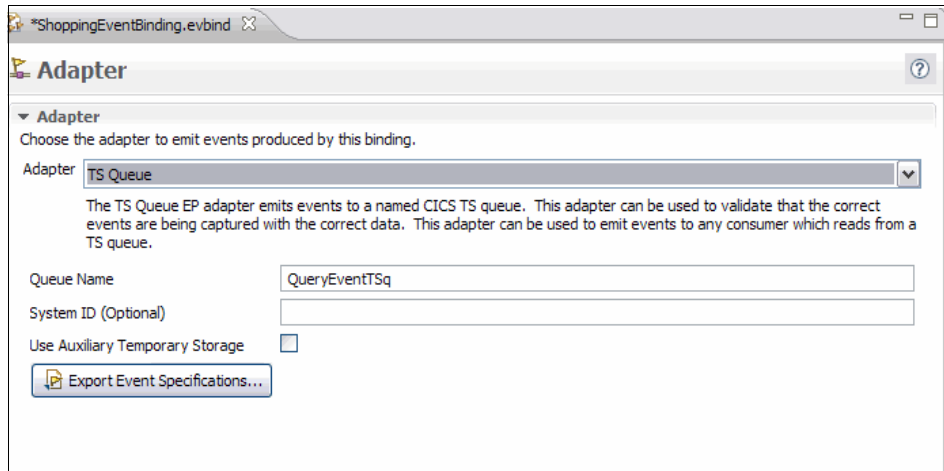


Figure 5-19 Specifying the Adapter

5.4 Exporting the event specification

Click **Export Event Specification**, in the Export Event Specification view, shown in Figure 5-20. Select the box next to QueryStock. Enter C:\ExportedEventSpecifications in the To directory box and click **Export**.

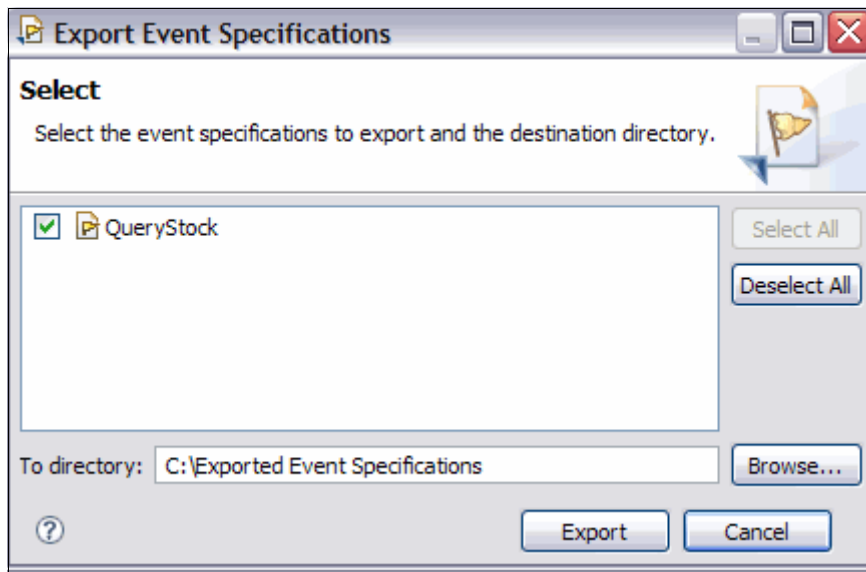


Figure 5-20 Export Event Specifications

The exported Event Specification is shown in Example 5-2. The exported copybook can be used for inclusion in a COBOL program written to read the temporary storage queue.

Note: In this release of CICS TS the exported Event Specification for a TS Queue adapter will always be a COBOL copybook.

Example 5-2 Exported Event Specification

```
* Generated copybook for Event Specification
* 'QueryStock'
01 QueryStock.
   05 ContextData.
      COPY DFHEPFEO.
   05 EventData.
      10 CustomerNumber          PIC +9(5) .
      10 StockId                 PIC +9(5) .
```

To map the context data in the context container, the following copybooks are shipped with CICS:

- ▶ DFHEPFEO: COBOL
- ▶ DFHEPFED: Assembler
- ▶ DFHEPFEL: PL/I
- ▶ DFHEPFEH: C

5.5 Exporting the bundle project

To use the event specifications in CICS TS we first export the bundle project. We export the bundle direct to a System z HFS, where CICS can read it directly.

In the project view of IBM CICS Explorer, right-click the **ShoppingEventBundle** project and click **Export to System z HFS** as shown in Figure 5-21.

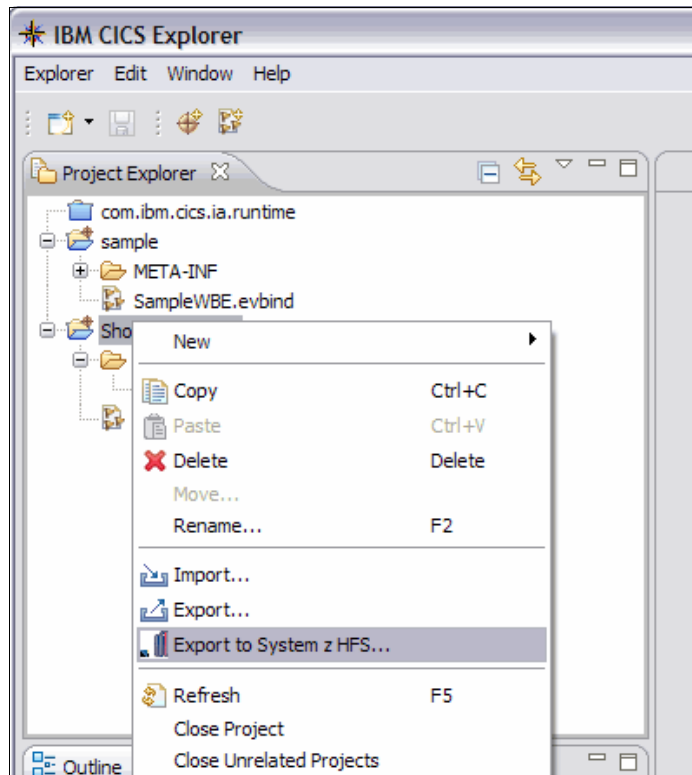


Figure 5-21 Export the ShoppingEventBundle

In the **Export to System z HFS** view, select the **ShoppingEventBundle** project and specify the following values in the Destination system details:

- ▶ Host: sc66ts
- ▶ FTP Port: 21
- ▶ Username: CICSRS2
- ▶ Password: CICSRS2's password
- ▶ HFS directory: /u/cicsrs2/bundles

Note: The Username must be a valid TSO user who has write access to the specified HFS directory.

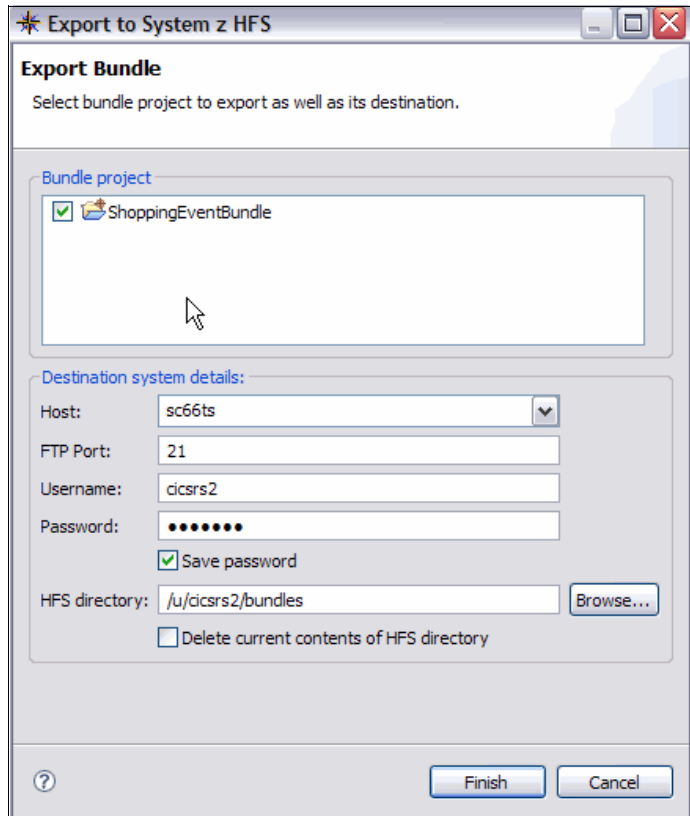


Figure 5-22 Export bundle to System z HFS

Note: As an alternative to Export to System z HFS, the bundle can be exported to the local file system, and a File Transfer Program can be used to copy the event binding file to System z HFS. The file transfer must be in binary.

After the export we have the file structure shown in Example 5-3 on page 112 on z/OS.

The export creates two files:

- ▶ /u/cicsrs2/bundles/ShoppingEventBundle/META-INF/cics.xml
Shown in Example 5-4.
- ▶ /u/cicsrs2/bundles/ShoppingEventBundle/ShoppingEventBinding.evbind
Shown in Example 5-5.

Note: A directory name of ShoppingEventBundle is added by the Export function. The name comes from the CICS Explorer project name and must be used in the CICS resource definition.

Example 5-3 Exported bundle files

```
Dir /u/cicsrs2/bundles/ShoppingEventBundle/  
  Dir  META-INF  
    File  cics.xml  
    File  ShoppingEventBinding.evbind
```

The CICS bundle manifest created by the export is shown in Example 5-4.

Example 5-4 cics.xml file created by bundle export

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<manifest xmlns="http://www.ibm.com/xmlns/prod/cics/bundle" bundleVersion="1"  
bundleRelease="0" build="EUR">  
  <meta_directives>  
    <timestamp>2009-06-30T11:35:23.034+01:00</timestamp>  
  </meta_directives>  
  <define name="ShoppingEventBinding"  
type="http://www.ibm.com/xmlns/prod/cics/bundle/EVENTBINDING"  
path="/ShoppingEventBinding.evbind"/>  
</manifest>
```

The exported event binding file is shown in Example 5-5.

Example 5-5 ShoppingEventBinding.evbind file created by bundle export

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>  
<ns2:eventBinding CICSEPSchemaRelease="0" CICSEPSchemaVersion="1"  
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/cics/eventprocessing/eventbin  
ding CicsEventBinding.xsd "  
xmlns:ns2="http://www.ibm.com/xmlns/prod/cics/eventprocessing/eventbinding"  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">  
  <description>The Shopping event binding.</description>  
  <userTag></userTag>  
  <eventSpecification>
```

```

<name>QueryStock</name>
<description>Query stock event specification</description>
<eventInformation>
  <eventInformationItem description="The customer number" length="6"
dataPrecision="0" dataType="numeric" name="CustomerNumber"/>
  <eventInformationItem description="The stock identification"
length="6" dataPrecision="0" dataType="numeric" name="StockId"/>
</eventInformation>
</eventSpecification>
<eventCaptureSpecification>
  <name>CaptureQueryEvent</name>
  <eventIdentifier>QueryStock</eventIdentifier>
  <description>Capture the Query Events</description>
  <filter>
    <contextFilter>
      <transactionId filterValue="" filterOperator="OFF"/>
      <currentProgram filterValue="" filterOperator="OFF"/>
      <userId filterValue="" filterOperator="OFF"/>
      <CommandResp filterValue="OK" filterOperator="OFF"/>
      <EIBAIID value="" filterOperator="OFF"/>
      <EIBCPASN filterValue="1" filterOperator="OFF"/>
    </contextFilter>
    <locationFilter filterType="PROGRAM_INIT">
      <programInit>
        <PROGRAM filterValue="QUERY" filterOperator="EQ"
keyword="PROGRAM"/>
        <CHANNEL filterValue="" filterOperator="OFF"
keyword="CHANNEL"/>
      </programInit>
    </locationFilter>
    <dataFilter>
      <filterItem>
        <dataFilter filterValue="" filterOperator="EXS"
languageVariableName="" dataPrecision="0" dataType="CHAR" length="1" offset="0"
container="DISP-STOCK-ITEM" source="CHANNEL"/>
      </filterItem>
    </dataFilter>
  </filter>
  <dataCapture>
    <captureItem>
      <dataCaptureItem formatPrecision="0" formatdataType="numeric"
formatlength="6" eventItemName="CustomerNumber" languageVariableName=""
captureDataPrecision="0" captureDataType="PACKED" captureLength="3" offset="0"
container="CUSTOMER-NO" source="CHANNEL"/>
    </captureItem>
    <captureItem>
      <dataCaptureItem formatPrecision="0" formatdataType="numeric"
formatlength="6" eventItemName="StockId" languageVariableName=""

```

```

captureDataPrecision="0" captureDataType="PACKED" captureLength="3" offset="0"
container="DISP-STOCK-ITEM" source="CHANNEL"/>
    </captureItem>
</dataCapture>
</eventCaptureSpecification>
<eventDispatcherSpecification>
    <eventDispatcher>
        <eventDispatcherPolicy>
            <dispatchPriority>normal</dispatchPriority>
            <eventsTransactional>false</eventsTransactional>
            <adapterUserid useContextUserid="false"></adapterUserid>
            <adapterTranId></adapterTranId>
        </eventDispatcherPolicy>
        <eventDispatcherAdapter>
            <cicsTSQueueAdapter>
                <queueName>QueryEventTSq</queueName>
                <sysid></sysid>
                <useAuxTempStorage>false</useAuxTempStorage>
                <format>CFE</format>
            </cicsTSQueueAdapter>
        </eventDispatcherAdapter>
    </eventDispatcher>
</eventDispatcherSpecification>
</ns2:eventBinding>

```

5.6 Installing the bundle in CICS

To use the Event Binding definition in CICS, create two CICS resource definitions:

- ▶ Resource Group Definition
- ▶ Bundle Definition

We use CICS Explorer to create the resources. In CICS Explorer we open the CICS SM perspective (Figure 5-23 on page 115).

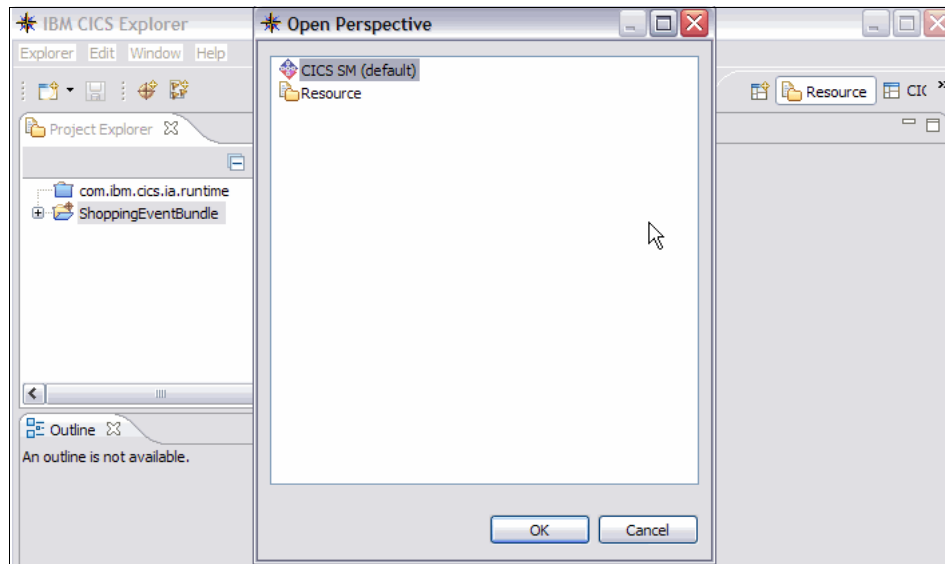


Figure 5-23 CICS Explorer Open Perspective

1. Perform the following steps to first create the Resource Group.
 - a. Click **Administration Resource Group Definitions** and right-click the white space.
 - b. Click **New** and enter the following information (Figure 5-24). Click **Finish**:
 - Data Repository: EPRED
 - Name: EPGRP2
 - Description: The Resource Group

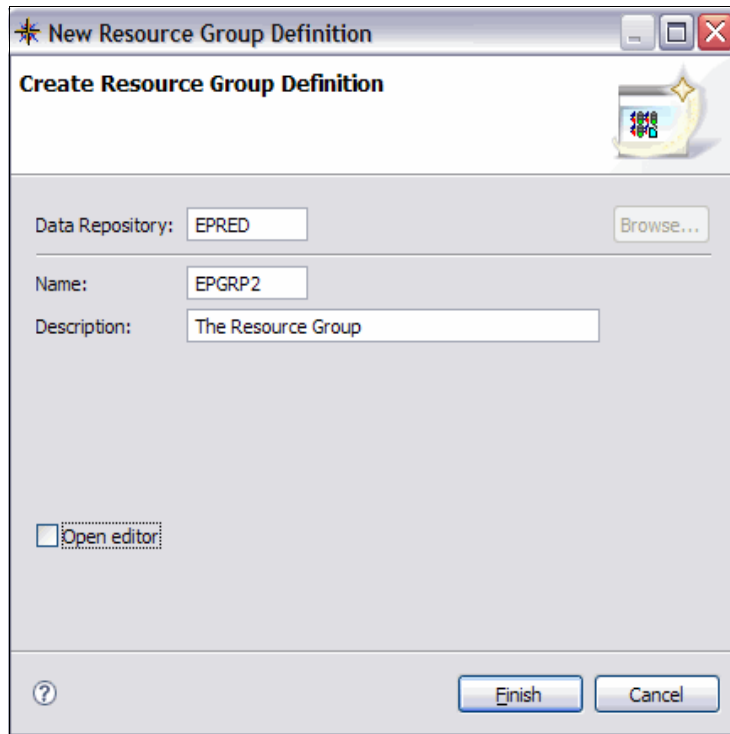


Figure 5-24 Create the Resource Group

- c. Click **Administration Bundle Definitions**. In the presented view, right-click the white space, click **New**, and enter the information:
 - Data Repository: EPRED
 - Resource Group: EPGRP2
 - Name: SHOPEVE
 - Description: The Shopping Bundle definition
 - Bundle Directory: /u/cicsrs2/bundles/ShoppingEventBundle
- Click **Finish**.

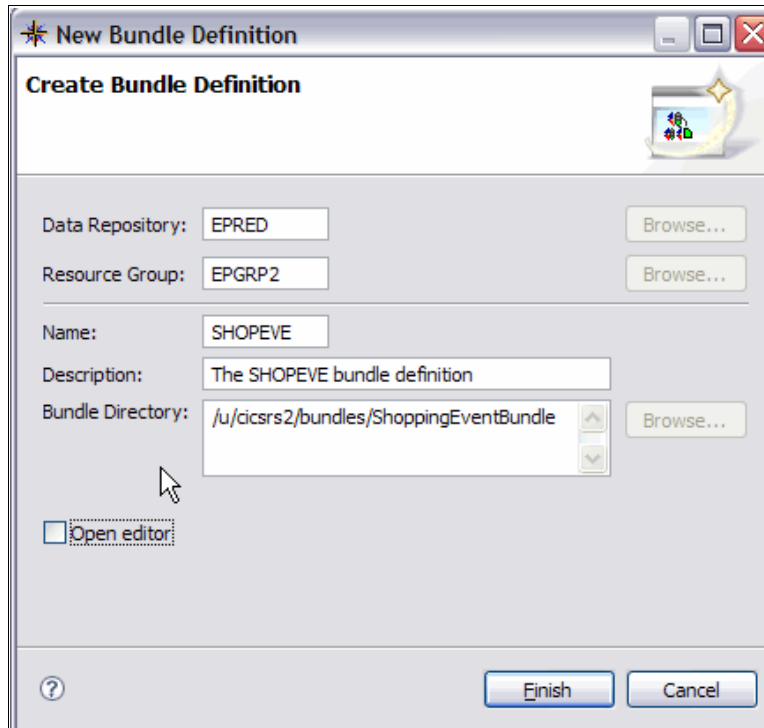


Figure 5-25 Create the BUNDLE resource definition

2. Install the SHOPEVE bundle resource. In the Bundle Definitions view (Figure 5-26) right-click the SHOPEVE bundle and click **Install**.

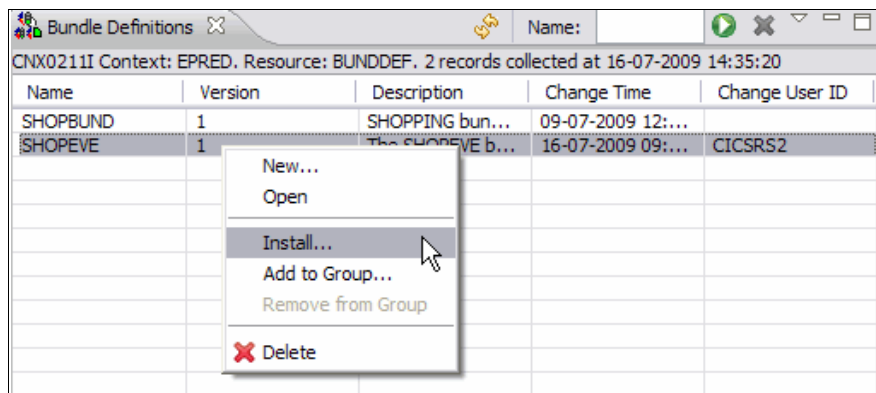


Figure 5-26 Installing the SHOPEVE bundle resource

3. In the window (Figure 5-27) check the target region for the install and click **OK**.

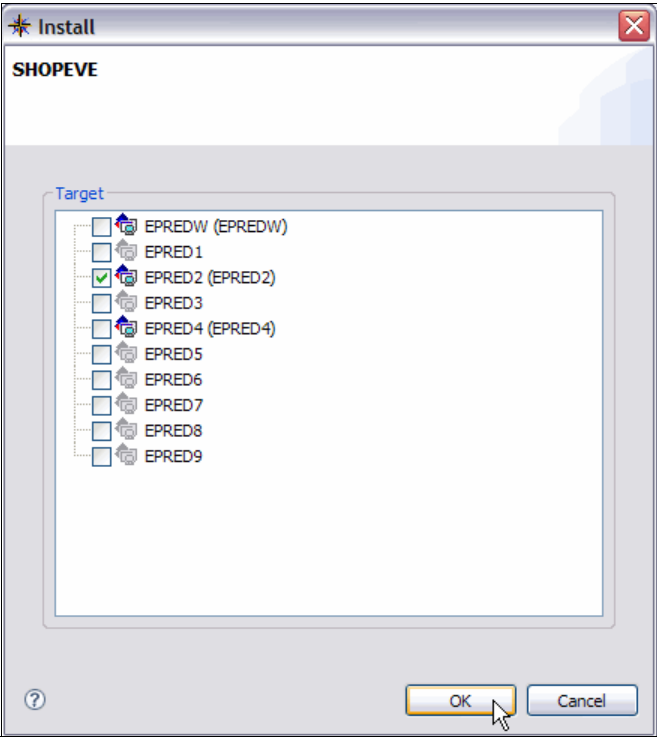


Figure 5-27 Select target region for install

4. Verify the install. In CICS Explorer, click **Operations Bundles**. Figure 5-28 shows the installed SHOPEVE bundle resource.

The image shows a screenshot of the "Bundles" view in CICS Explorer. The window title is "Bundles". Below the title bar, there is a status bar that reads "CNX0211I Context: EPRED2. Resource: BUNDLE. 1 records collected at 16-07-2009 14:39:10". Below the status bar is a table with four columns: "Region", "Name", "Status", and "Install Time". The table contains one row of data.

Region	Name	Status	Install Time
EPRED2	SHOPEVE	✓ ENABLED	16-07-2009 09:39:01

Figure 5-28 CICSplex Explorer Bundle view

5.7 Testing the Shopping application

Now we test to see whether a customer query on a stock item in our shopping application causes a QueryStock event to be emitted.

1. On a CICS terminal enter the transaction ID MENU and press **Enter**. On the following panel (Figure 5-29), enter the Customer Number 00005 and press **PF1** to query an item.

```
Order application

Customer Number: . . . . . 00005

PF1   Query Item
PF2   Order Item
PF4   fulfill Order
PF5   Ship Order

Exit = PF3
```

Figure 5-29 The Shopping Application MENU panel

2. On the following panel (Figure 5-30) enter S in front of ID 00006 and press **Enter**.

```
Query List
Customer Number: 00005

S ID      Description
- - - - -
00001 STOCK1
00002 STOCK2
00003 STOCK3
00004 STOCK4
00005 STOCK5
S 00006 STOCK6
```

Figure 5-30 MENU query stock

The result panel is shown in Figure 5-31.

Query Item
Customer Number: 00005
Stock ID: 00006
Description: STOCK6
Value: 6.99
Stock Level: 00600

Figure 5-31 The Query Stock result panel

5.8 Verifying the event processing

We use CICSplex Explorer to verify event processing.

1. Select CICS region EPRED2 within the CICSplex. Click **Operations Event Processing Event Processing** (Figure 5-32).

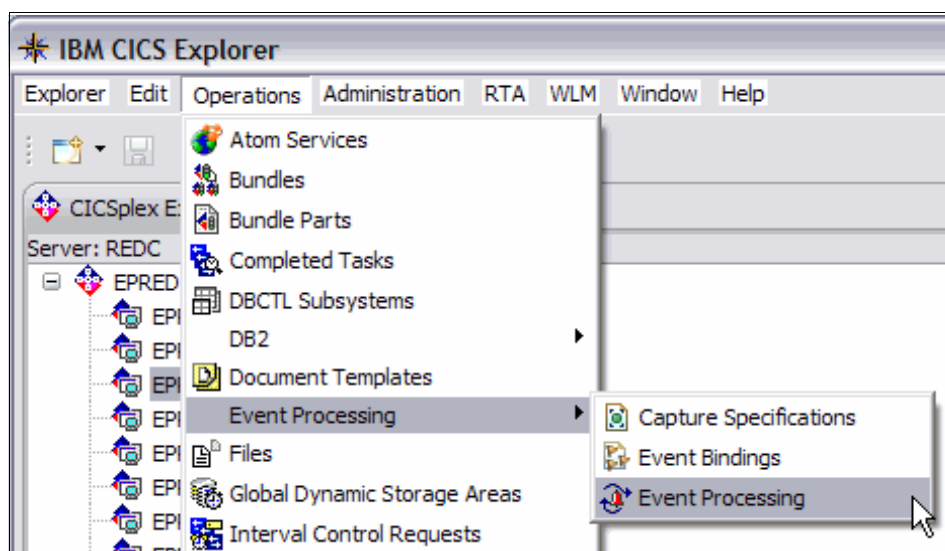


Figure 5-32 Monitor event processing

Column Put Events (Figure 5-33) shows one event has been emitted.

Note: The CICS supplied transaction CEBR can be used to display the queue too. The command looks like this:

CEBR QueryEventTSq

Example 5-7 shows the data put to Temporary Storage queue, QueryEventTSq, mapping to the exported Event Specification (Example 5-2 on page 109). The first part is CICS supplied copybook DFHEPFEO and the second part are the emitted business information elements.

Example 5-7 Data put to the Temporary Storage queue

EPFEIDENTIFIER	PIC X(4)	EPFE	Identifier
EPFEVERSION	PIC X(4)	0001	Version
EPFEEVENTBINDING	PIC X(32)	ShoppingEventBinding	Event Binding Name
EPFEEBUSERTAG	PIC X(8)		Event Binding user tag
EPFEBUSINESSEVENT	PIC X(32)	QueryStock	Business event name
EPFENETWORKUOWID	PIC X(54)	1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F2F974430CF4C10D000100	Network UOW ID
EPFENETQUALAPPLID	PIC X(17)	USIBMSC.EPRED2	Network Applid Qualifier Applid
EPFEDATETIME	PIC X(25)	2009-07-08T16:04:59+00:00	Date Time
EPFECAPSPECNAME	PIC X(32)	CaptureQueryEvent	Capture Specification Name
FILLER	PIC X(16)		Reserved
Event data starts here			
CustomerNumber	PIC +9(5)	+00005	
StockId	PIC +9(5)	+00006	

5.9 Creating events to the Transaction Start EP adapter

Having tested the event specification using the Temporary Storage Queue EP adapter, we change the event specification to use the Transaction Start EP adapter.

1. In CICS Explorer, open the Resource perspective. In the Project Explorer window, expand the ShoppingEventBundle project. Right-click ShoppingEventBinding.evbind and click **Open**. In the presented window (Figure 5-36), click the **Adapter** tab at the bottom of the window.

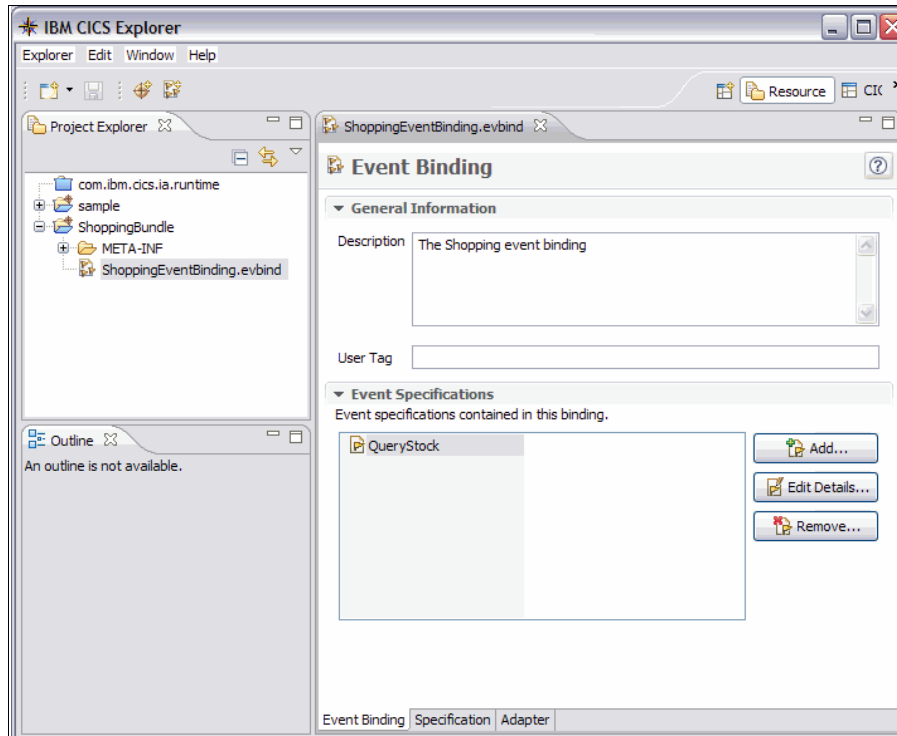


Figure 5-36 Changing the adapter

2. On the next window (Figure 5-37) set the following values:

- Adapter: Transaction Start
- Transaction: IDSNIF

Press **Ctrl-S** to save the change.

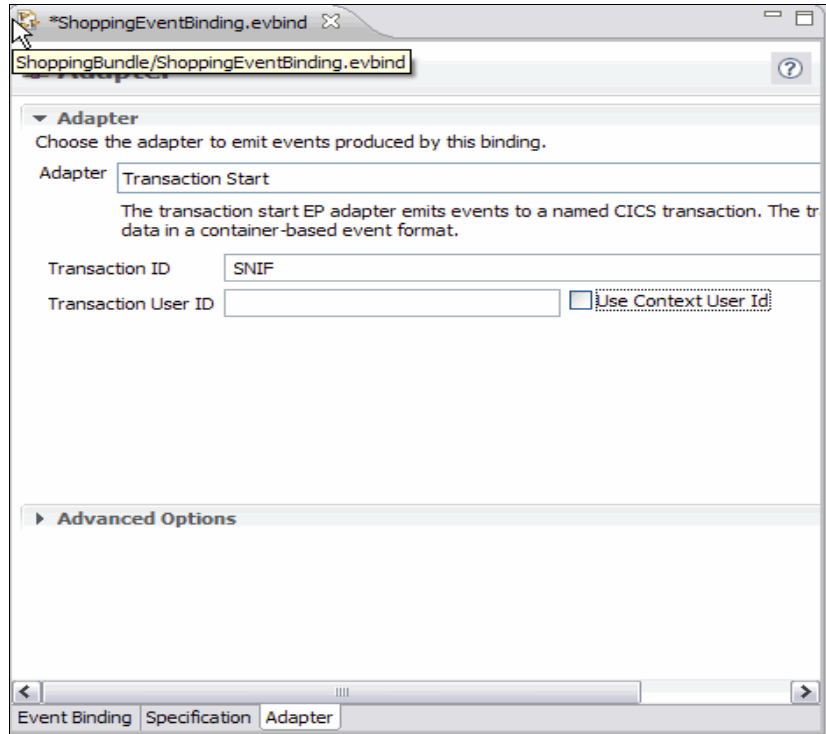


Figure 5-37 Transaction Start adapter

3. Discard the BUNDLE resource in CICS, and repeat the steps in 5.5, "Exporting the bundle project" on page 110 through 5.7, "Testing the Shopping application" on page 119. The changed part of the event binding file is shown in Figure 5-38 on page 125.

Note: Once a Transaction ID has been defined for the EP Adapter, it will be kept in the event binding until manually deleted. To delete the Transaction ID the Advanced Options must be expanded, and the Transaction ID field cleared. If the field is not blank, the transaction ID specified will be attached, the associated program will be executed, and the defined adapter type has no effect. The same can be achieved by removing the transaction ID from the exported event binding file.

```
<adapterTranId>SNIF</adapterTranId>
```

```
<eventDispatcher>
  <eventDispatcherPolicy>
    <dispatchPriority>normal</dispatchPriority>
    <eventsTransactional>false</eventsTransactional>
    <adapterUserid useContextUserid="false"></adapterUserid>
    <adapterTranId></adapterTranId>
  </eventDispatcherPolicy>
  <eventDispatcherAdapter>
    <cicsTransactionAdapter>
      <transactionId>SNIF</transactionId>
      <sysid></sysid>
      <userid useContextUserid="true"></userid>
      <format>CCE</format>
    </cicsTransactionAdapter>
  </eventDispatcherAdapter>
</eventDispatcher>
```

Figure 5-38 CICS Transaction EP adapter binding file

We show the content of the channel passed to the attached transaction in Figure 5-39.

In our test the attached transaction, SNIF, is running a program SNIFFY. This program does a browse of the passed channel, and puts the container names and contents to the CESE transient data queue.

```
*** Start ***
Containers on channel: List starts.
>=====<
Container Name   : DFHEP.CCECONTEXT
Content length  : 00000224
Container content: EPFE0001ShoppingEventBinding
QueryStock
                1910E4E2C9C2D4E2C34BE3
C3D7F6F6F0F2F77A71762AA9DB000100USIBMSC.EPRED2    2009-07-13T14:04:34+00:00
CaptureQueryEvent
>=====<
Container Name   : DFHEP.NAME.00001
Content length  : 00000032
Container content: CustomerNumber
>=====<
Container Name   : DFHEP.DATA.00001
Content length  : 00000006
Container content: +00005
>=====<
Container Name   : DFHEP.NAME.00002
Content length  : 00000032
Container content: StockId
>=====<
Container Name   : DFHEP.DATA.00002
Content length  : 00000006
Container content: +00006
Containers on channel: List ends
**** End ****
```

Figure 5-39 Container content for Transaction Start EP adapter written to CESE.

5.10 Creating event to WebSphere Business Events

Next, test emitting the same event to WebSphere Business Event.

1. In CICS Explorer open the Resource perspective. In the Project Explorer window, expand the ShoppingBundle project. Right-click ShoppingEventBinding.evbind and click **Open**. In the presented window (Figure 5-36 on page 123), click the **Adapter** tab at the bottom of the window. Set the following values:

- Adapter: WMQ Queue
- Queue Name: SHOPPING_EVENT_QUEUE
- Data Format: WebSphere Business Events (XML)

Press **Ctrl-S**.

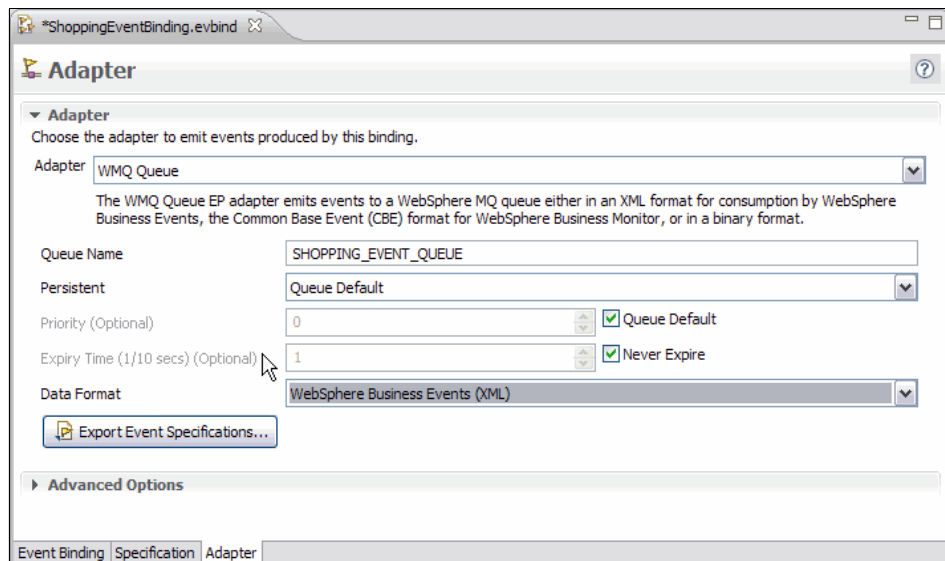


Figure 5-40 Changing adapter type

We show the WebSphere Business Events EP adapter specifics from the event binding file in Figure 5-41.

```

<eventDispatcherSpecification>
  <eventDispatcher>
    <eventDispatcherPolicy>
      <dispatchPriority>normal</dispatchPriority>
      <eventsTransactional>false</eventsTransactional>
      <adapterUserid useContextUserid="false"></adapterUserid>
      <adapterTranId></adapterTranId>
    </eventDispatcherPolicy>
    <eventDispatcherAdapter>
      <wmqAdapter>
        <queueName>SHOPPING_EVENT_QUEUE</queueName>
        <persistent>QUEUE_DEFAULT</persistent>
        <priority>-1</priority>
        <expiryTime>-1</expiryTime>
        <format>WBE</format>
      </wmqAdapter>
    </eventDispatcherAdapter>
  </eventDispatcher>
</eventDispatcherSpecification>

```

Figure 5-41 WebSphere Business Event EP adapter specifics

- Discard the BUNDLE resource in CICS, and perform the steps in 5.5, “Exporting the bundle project” on page 110 through 5.7, “Testing the Shopping application” on page 119.
- We use IBM WebShere MQ Explorer(Figure 5-42) to show the message put to the WMQ queue by the adapter.We see one message put to the queue, SHOPPING_EVENT_QUEUE.

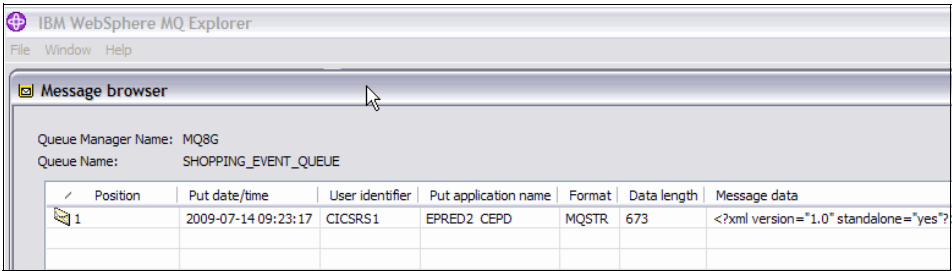


Figure 5-42 The message emitted to the WMQ queue

- Right-click the line showing the message -and click Properties (Figure 5-43 on page 129).

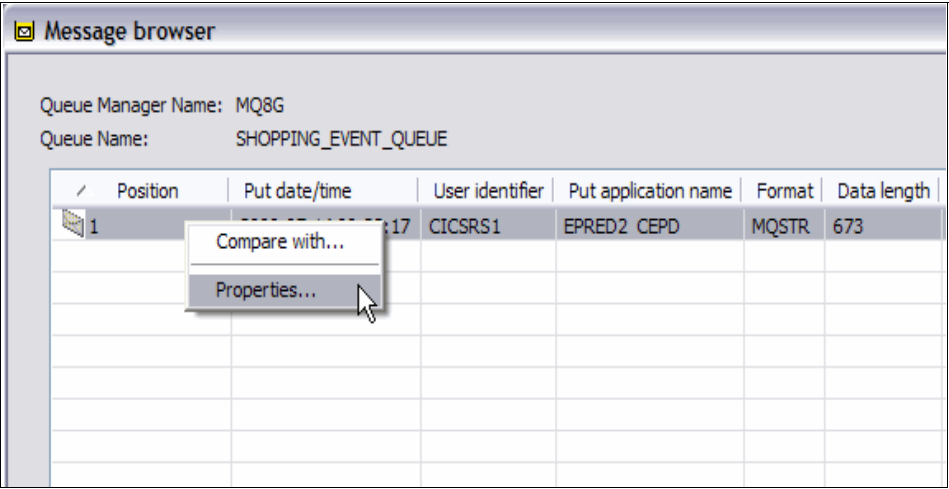


Figure 5-43 WebSphere MQ message browser

4. In the properties window, click **Data** (Figure 5-44). We see the message shown in character and hex format.

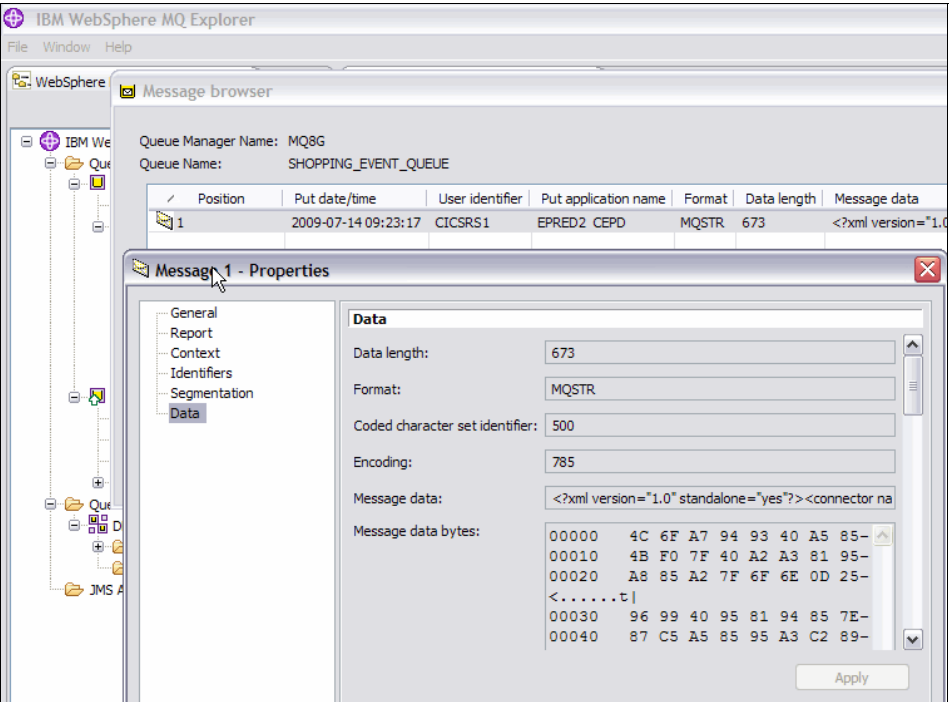


Figure 5-44 WebSphere MQ Explorer data view

For the purpose of the book we mark Message data and copy it to Figure 5-45.

```
<?xml version="1.0" standalone="yes"?>
<connector name="ShoppingEventBinding" version="2.2">
  <connector-bundle name="QueryStock" type="Event">
    <connector-object name="QueryStock_Context">
      <field name="Binding user tag"></field>
      <field name="Network
UOWID">1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F1F676AA6564BD44000100</field>
      <field name="Capture Spec Name">CaptureQueryEvent</field>
    </connector-object>
    <connector-object name="QueryStock_Data">
      <field name="CustomerNumber">+00005</field>
      <field name="StockId">+00006</field>
    </connector-object>
  </connector-bundle>
</connector>
<system>USIBMSC.EPRED2</system>
  <timestamp>2009-07-10T13:58:00+00:00</timestamp>
</connector>
```

Figure 5-45 WMQ message sent to WebSphere Business Events

Having tested that the event in WebSphere Business Events format has been written to the WebSphere MQ queue, we set up WebSphere Business Events to receive this event. We show this in Chapter 7, “WebSphere Business Events scenario” on page 183.

5.11 Adding event specifications

In this procedure, we add more event specifications to the existing event binding.

1. In the IBM CICS Explorer, open the **Resource Perspective** and expand the **ShoppingEventBundle**. Double-click ShoppingEventBinding.evbind. This opens the Event Binding edit window (Figure 5-46 on page 131). Click **Add** next to the Event Specification part of the window.

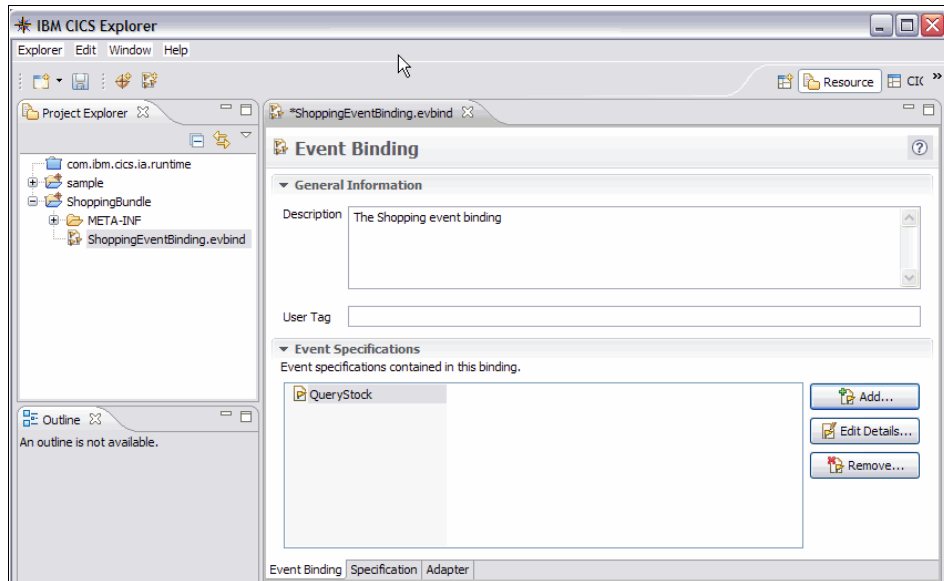


Figure 5-46 Add event specification

2. In the “Add Event Specification” window, enter (Figure 5-47) the following values:
 - Name: Order
 - Description: Order event specification
 Click **OK**.

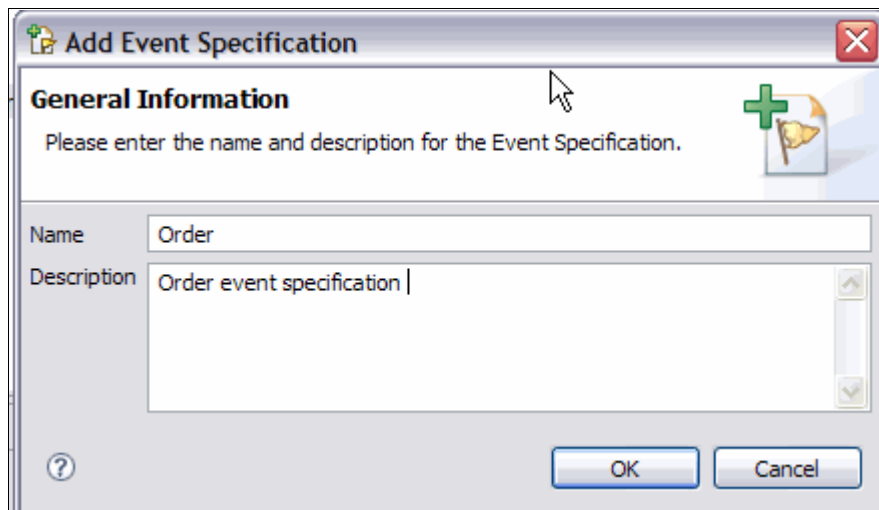


Figure 5-47 Add Order event specification

3. Click the **Order** specification and the **Specification** tab and add the Emitted Business Information as shown in Table 5-1 and Figure 5-48.

Table 5-1 Emitted business information

Name	Type	Length	Precision	Description
CustomerNumber	Numeric	6	0	The customer number
OrderNumber	Numeric	6	0	The order number
StockId	Numeric	6	0	The stock ID
Quantity	Numeric	6	0	The quantity

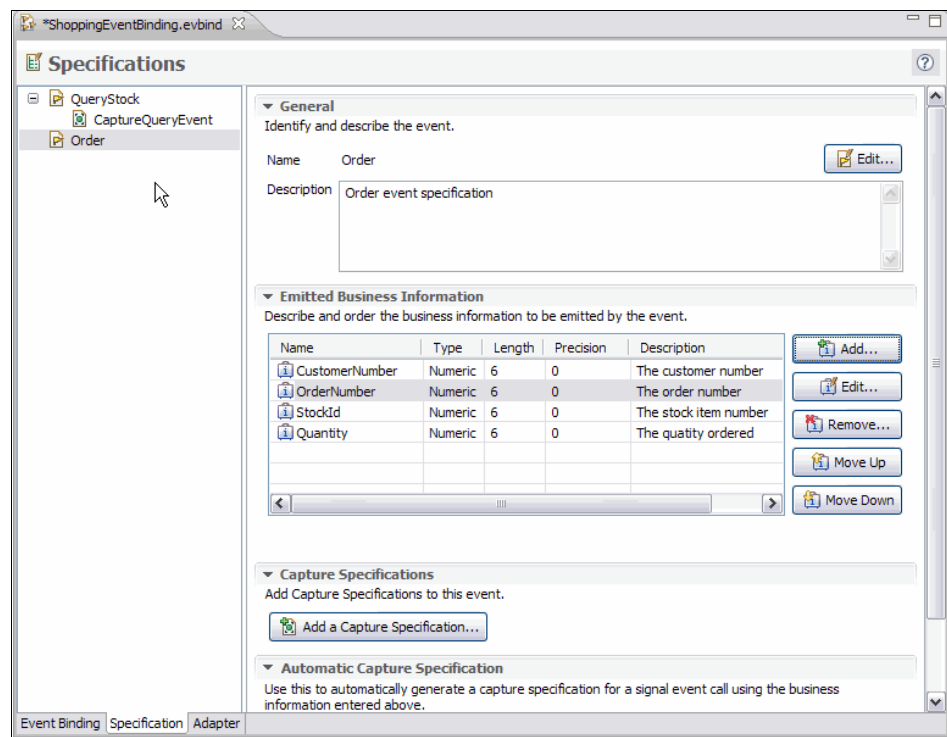


Figure 5-48 Adding business information

4. Click **Add a Capture Specification**. In the “Add Capture Specification” window (Figure 5-49 on page 133), enter the following values:
 - Name: CaptureOrder
 - Description: Capture order specificationClick **OK**.

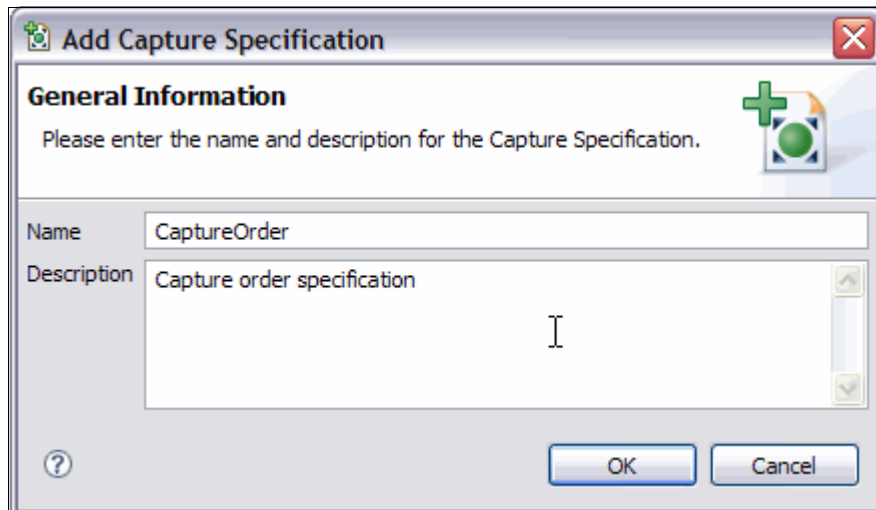


Figure 5-49 Capture Order specification

5. Click the **CaptureOrder** specification. On the Capture Point panel (Figure 5-50), set the Capture Point to LINK PROGRAM and make sure the **Capture after** radio button is selected. Click **Next Filtering**.

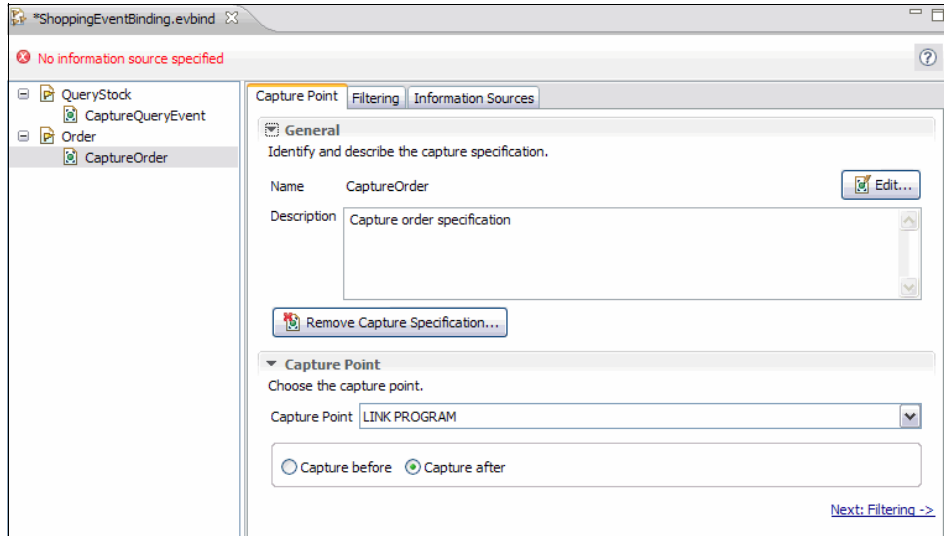


Figure 5-50 Set capture point

6. On the filtering panel in the Application Command Options part, set **PROGRAM*** with the following values:
 - Operator: Equals
 - Value: SENDORDR

Click **Next Information Sources**.

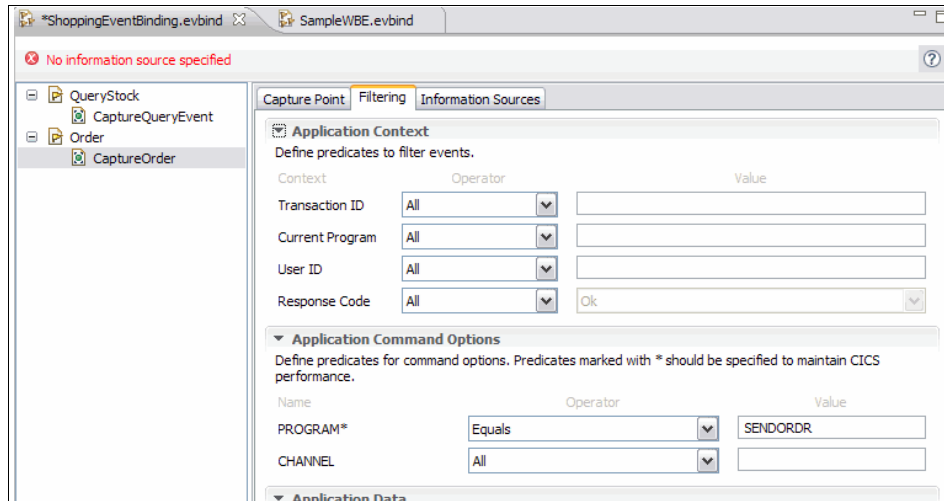


Figure 5-51 Set filtering

7. On the Information Sources panel, click the Information Sources one-by-one and click **Edit**. On the Information Source panel, click CHANNEL and enter the information shown in Table 5-2. The result is shown in Figure 5-52 on page 135.

Table 5-2 CaptureOrder information sources

Business information	Container	Type	Offset	Length	Precision
CustomerNumber	EPRED-CONTAINER3	Zoned Decimal	0	6	0
OrderNumber	EPRED-CONTAINER3	Packed Decimal	15	6	0
StockId	EPRED-CONTAINER3	Packed Decimal	5	6	0
Quantity	EPRED-CONTAINER3	Packed Decimal	10	6	0

Click **Ctrl-S** to save the work.

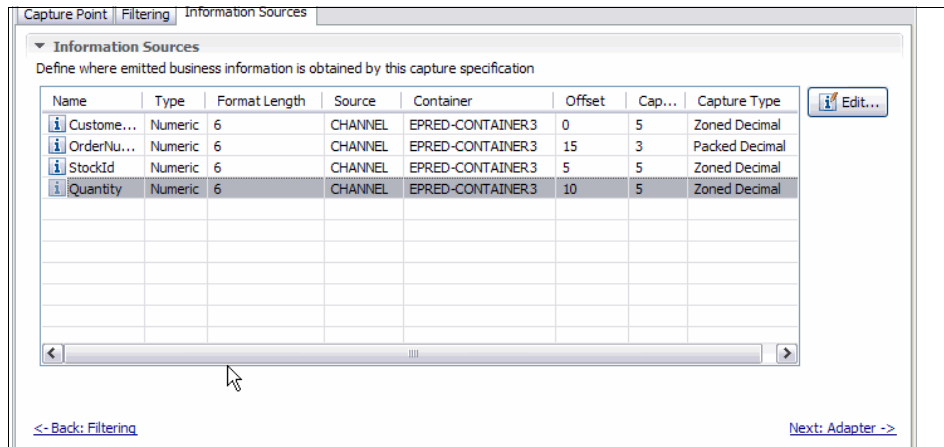


Figure 5-52 Setting information sources

8. Repeat the steps to add the following event specifications:
 - fulfill
See 5.11.1, “Fulfill event specification” on page 135.
 - Ship
See 5.11.2, “The Ship event specification” on page 139.
 - SendOrder
See 5.11.3, “The SendOrder event specification” on page 140.

5.11.1 Fulfill event specification

For the fulfill event specification, perform the following steps:

1. On the Add Event specification, enter the following values:
 - Name: fulfill
 - Description: fulfill event specification
2. On the “Emitted Business Information” panel, enter the values in Table 5-3.

Table 5-3 Fulfill emitted business information

Name	Type	Length	Precision	Description
StockId	Numeric	6	0	The stock ID
OldStock	Numeric	6	0	The old stock level
NewStock	Numeric	6	0	The new stock level

3. On the “Add Capture Specification” panel, enter the following values:
 - Name: Capturefulfill
 - Description: Capture fulfill specification
4. On the “Capture Point” panel, set the Capture Point to **PUT CONTAINER** and select the **Capture after** radio button.
5. On the “Filtering” panel, in the Application Context part, enter the following values:
 - Context: Current Program
 - Operator: Equals
 - Value: UPDSTOCK
6. In the Application Command Options part, enter the following values:
 - Name: CONTAINER*
 - Operator: Equals
 - Value: OUTPUT
7. In the Application Data part, enter the following values:
 - Source: From
 - Offset: 13
 - Length: 5
 - Operator: Less Than
 - Value: 50

Click **Next: Information Sources** on the “Information Sources” panel (Figure 5-53). Click **StockId** and **Edit**.

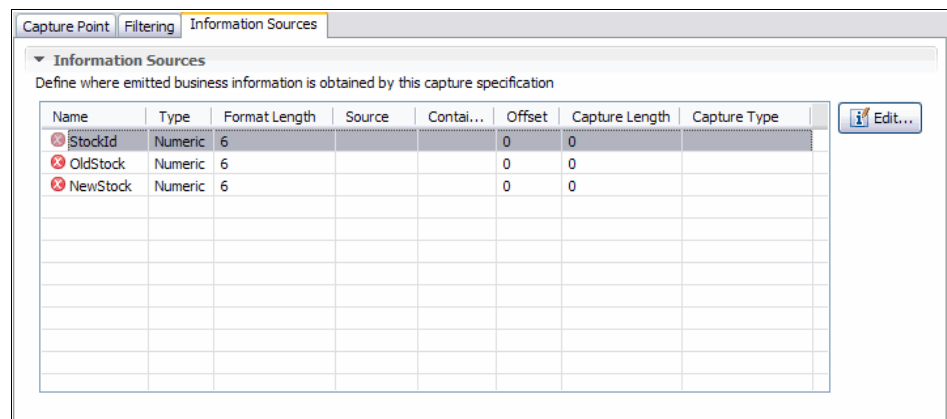


Figure 5-53 Information Sources panel

8. On the “Edit Information Source” panel (Figure 5-54 on page 137), click **Application data FROM** and then click **Select from imported language structure**.

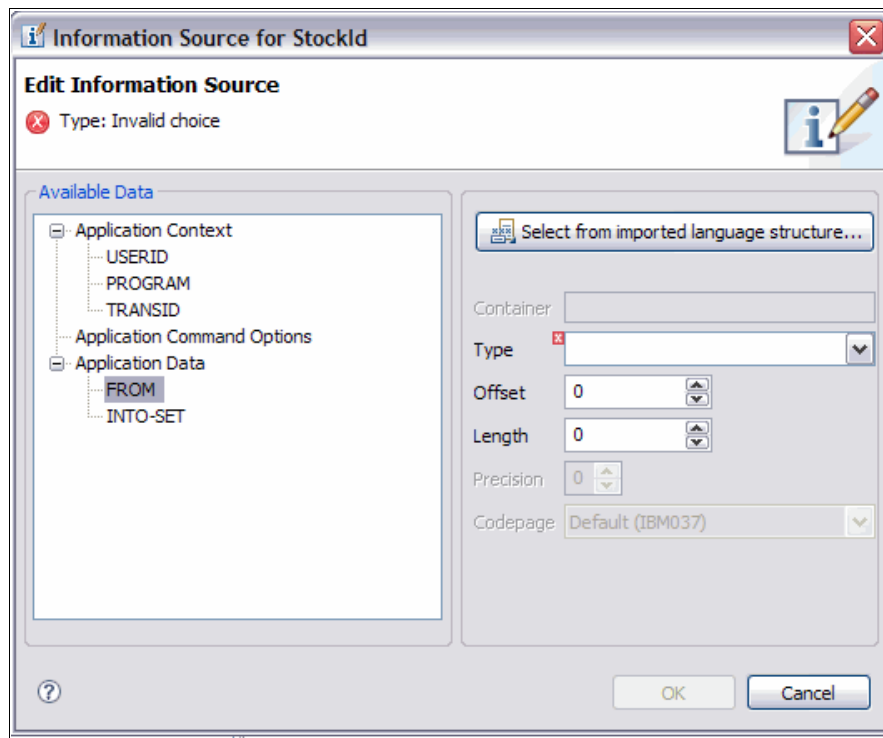


Figure 5-54 Import language structure

9. In the “Choose Source Code” panel (Figure 5-55), select the **COBOL** radio button. Click **Choose Language Structure File**. Select the COBOL copybook **UPDSTOCO**, and click **OK**.

Note: When using copybooks to map the emitted business information, they must be accessible from IBM CICS Explorer.

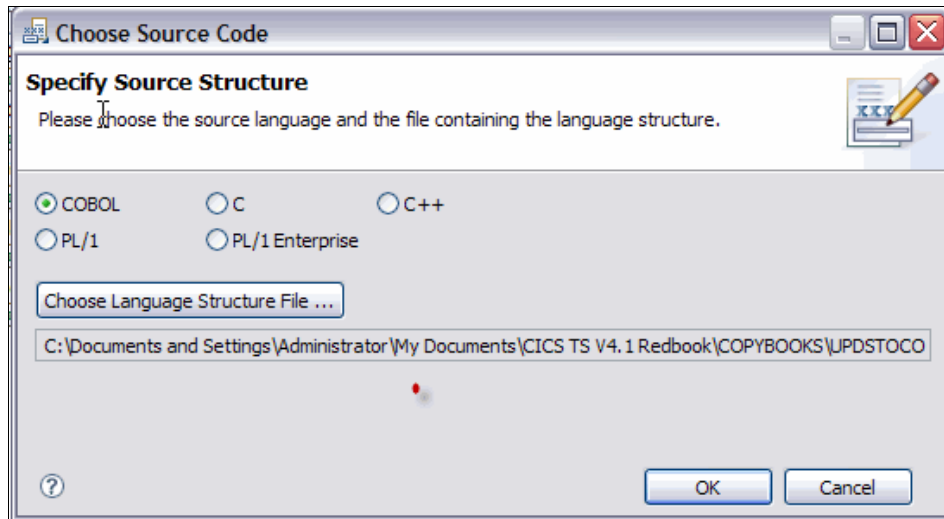


Figure 5-55 Specifying source structure

10. In the “Obtain data format from imported language structure” panel (Figure 5-56 on page 139), click **sid** then **OK**. Repeat the steps for **OldStock** and **NewStock**, where we set the following values:
 - OldStock: old-value
 - NewStock: new-valueClick **Ctrl-S** to save the changes.

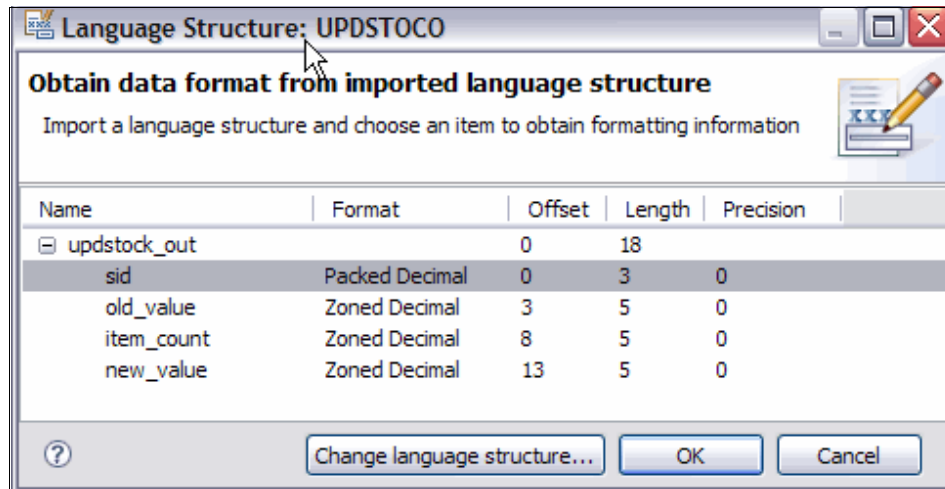


Figure 5-56 Define where to obtain the business information

5.11.2 The Ship event specification

Next, define the Ship event specification. Repeat all the steps in 5.11.1, “Fulfill event specification” on page 135 and enter the following information:

1. On the Add Event Specification, enter the following information:
 - Name: Ship
 - Description: Ship event specification

On the Emitted Business Information panel, specify the information shown in Table 5-4.

Table 5-4 Ship emitted business information

Name	Type	length	Precision	Description
OrderNumber	Numeric	6	0	The order number
CustomerNumber	Numeric	6	0	The customer number
StockId	Numeric	6	0	The stock ID
Quantity	Numeric	6	0	The order quantity

2. On the “Add Capture Specification” panel, specify the following information:
 - Name: CaptureShip
 - Description: Capture ship specification

3. On the “Capture Point” panel, set the Capture Point to **REWRITE** and select the **Capture after** radio button.
4. On the “Filtering” panel, in the Application Context part, set the following values:
 - Context: Current Program
 - Operator: Equals
 - Value: SHIP
 - Context: Response Code
 - Operator: Equals
 - Value: OK
5. In the Application Command Options part, set the following values:
 - Name: FILE*
 - Operator: Equals
 - Value: ORDER
 - Response Code
6. In the Application Data part we set:
 - Source: From
 - Offset: 75
 - Length: 1
 - Operator: Equals
 - Value: SClick **Next**.
7. In the Available Data part, click **FROM** and click **Select from imported language structure**. Select the **ORDER** copybook, and obtain the following data:
 - OrderNumber: OID
 - CustomerNumber: CID
 - StockId: SID
 - Quantity: ITEM-COUNTClick **Ctrl-S** to save the changes.

5.11.3 The SendOrder event specification

Next, we define the SendOrder event specification. We repeat all the steps in 5.11.1, “Fulfill event specification” on page 135 and enter the following information:

1. On the Add Event Specification:
 - Name: SendOrder
 - Description: SendOrder event specification

2. On the “Emitted Business Information” panel, specify the information shown in Table 5-5.

Table 5-5 *SendOrder emitted business information*

Name	Type	length	Precision	Description
CustomerNumber	Numeric	6	0	The order number
OrderNumber	Numeric	6	0	The customer number
CustomerName	Text	50	0	The stock ID
City	Text	50	0	The customer city
Country	Text	50	0	The customer country
Premium	Text	1	0	The customer type
TotalOrderValue	Numeric	11	0	Total order value

3. On the “Add Capture Specification” panel, specify the following values:
 - Name: CaptureSendOrder
 - Description: Capture SendOrder specification
4. On the Capture Point panel we set Capture Point to **EVENT***.
5. On the “Filtering” panel, in the Application Context part, set the following values:
 - Context: Current Program
 - Operator: Equals
 - Value: SENDORDRClick **Next**.
6. In Available Data, click **FROM** and click **Select from imported language structure**. Select the **SIGDATAO** copybook, and obtain the following data:
 - CustomerNumber: SIGNAL-CID
 - OrderNumber: SIGNAL-OID
 - CustomerName: SIGNAL-NAME
 - City: SIGNAL-CITY
 - Country: SIGNAL-COUNTRY
 - Premium: SIGNAL-PREMIUM
 - TotalOrdervalue: SIGNAL-TOTAL-VALUEClick **Ctrl-S** to save the changes.

5.12 Creating events for WebSphere Business Monitor

Next, we create an event binding file that will emit the events to WebSphere Business Monitor.

1. Open IBM CICS Explorer Resource perspective. In the Project Explorer window, right-click the **ShoppingEventBundle** and click **Copy** (Figure 5-57).

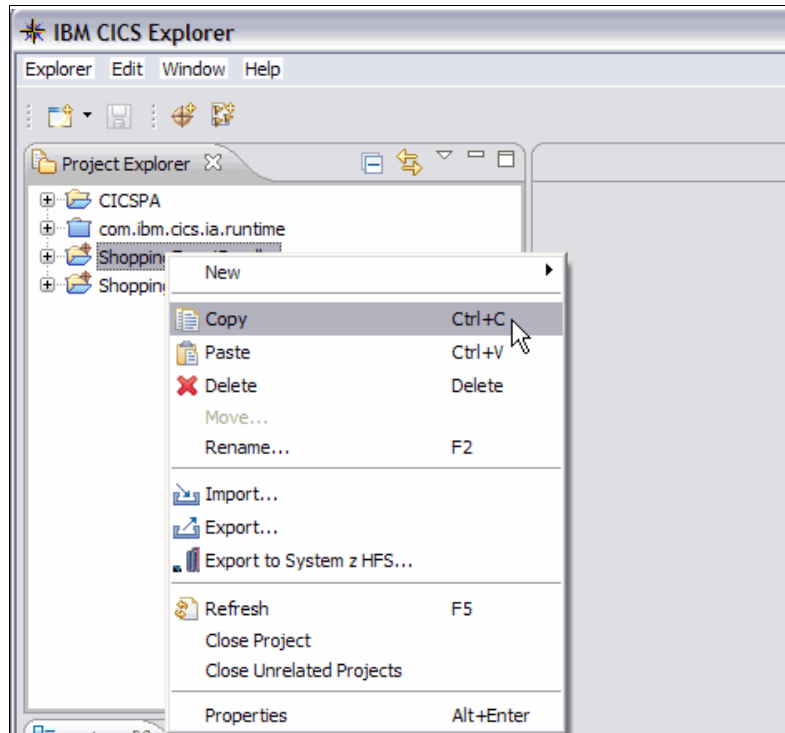


Figure 5-57 Copy ShoppingEventBundle project

2. Right-click the white space in the Project Explorer window and click **Paste**. Rename the project ShoppingMonitorBundle (Figure 5-58 on page 143).

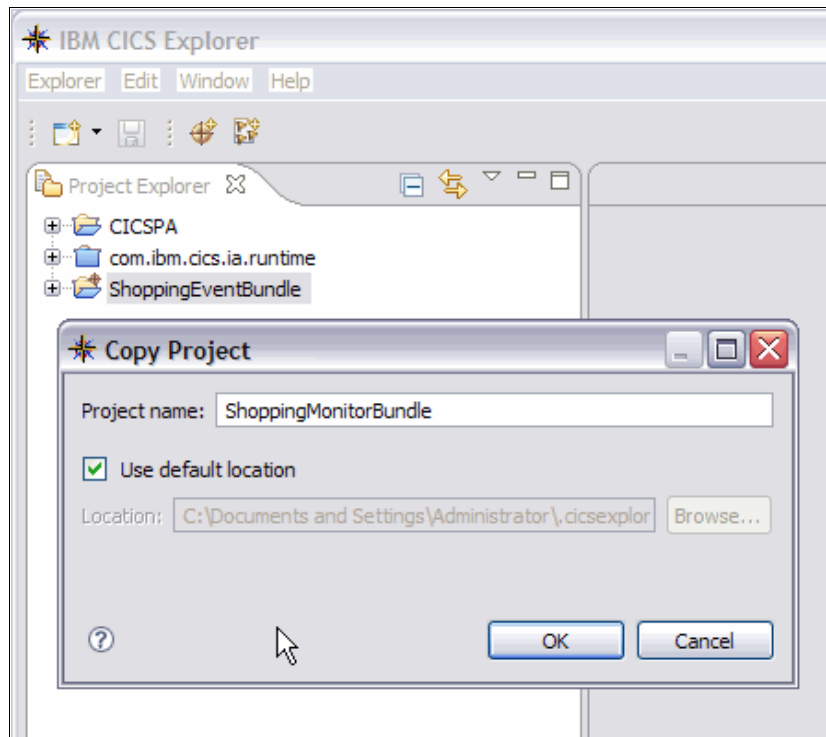


Figure 5-58 Create the ShoppingMonitorBundle

3. Change the name of the binding file. Double-click the **ShoppingMonitorBundle** project and right-click ShoppingEventBinding.evbind. Click **Rename** and change the name to ShoppingMonitorBinding.evbind.

4. Double-click the ShoppingMonitorBinding.evbind file to open the event binding. In the Event Specifications part of the Event Bindings window, click **Fulfill** and **Remove** to remove the fulfill specification from the event binding (Figure 5-59). Click **Ctrl-S** to save the changes.

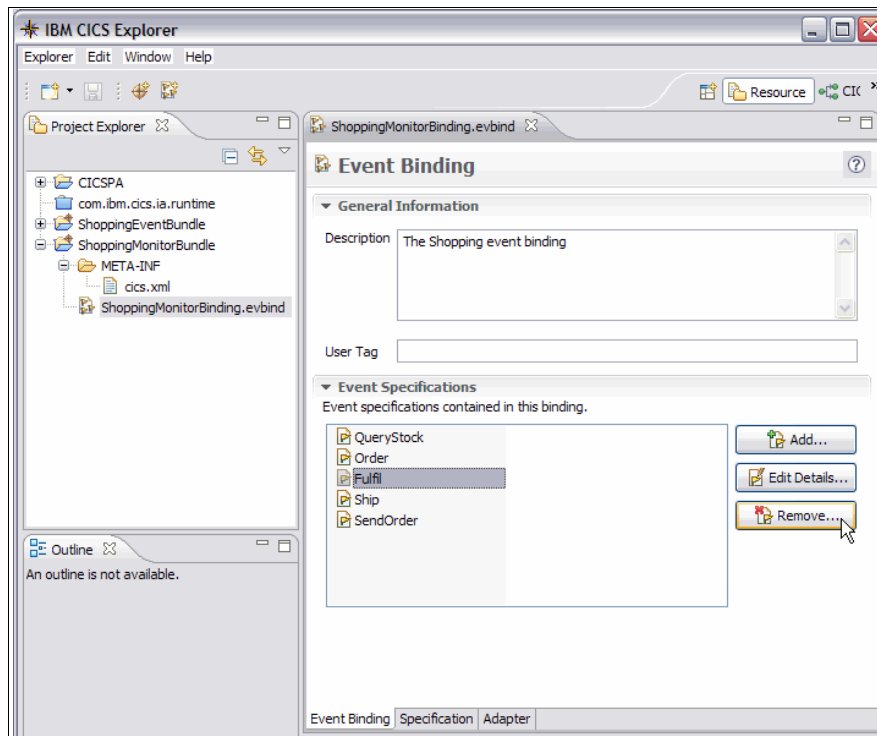


Figure 5-59 Remove fulfill event specification

5. Click the Adapter tab (Figure 5-60 on page 145) and set the following values:
 - Adapter: WMQ Queue
 - Queue Name: WBM.QUEUE
 - Data Format: Common Base Event (XML)Click **Ctrl-S** to save the changes.

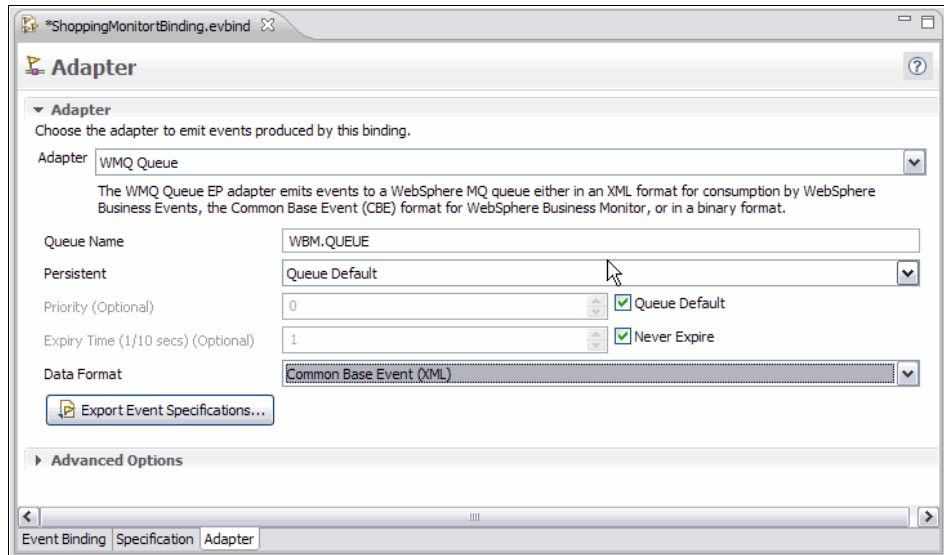


Figure 5-60 WebSphere Business Monitor adapter specifications

6. Click **Export Event Specifications** to save the event binding outside the IBM CICS Explorer workspace (Figure 5-61).

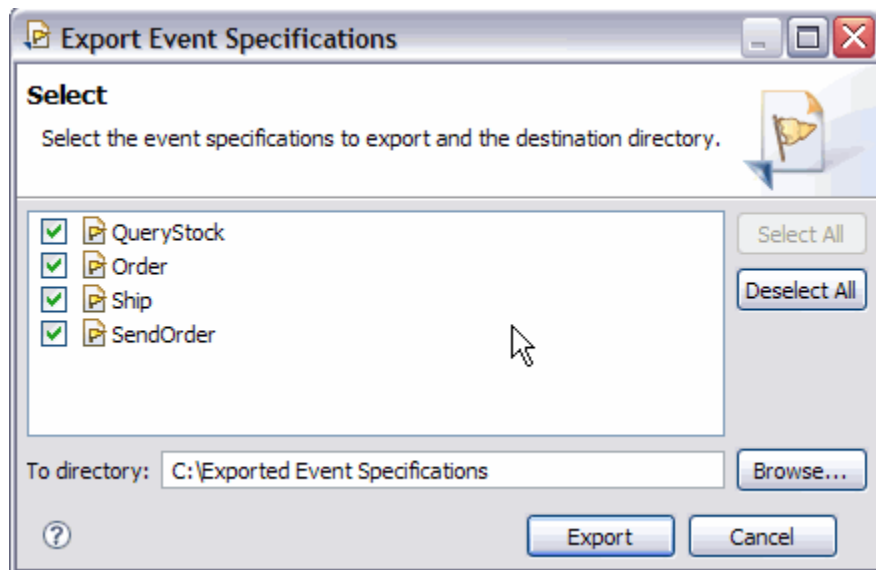


Figure 5-61 Export the ShoppingMonitorBinding specifications

Figure 5-62 displays the changes to the event binding file.

- queueName: WMB.QUEUE
- format: CBE

```
<eventDispatcherSpecification>
  <eventDispatcher>
    <eventDispatcherPolicy>
      <dispatchPriority>normal</dispatchPriority>
      <eventsTransactional>false</eventsTransactional>
      <adapterUserid
useContextUserid="false"></adapterUserid>
      <adapterTranId></adapterTranId>
    </eventDispatcherPolicy>
    <eventDispatcherAdapter>
      <wmqAdapter>
        <queueName>WMB.QUEUE</queueName>
        <persistent>QUEUE_DEFAULT</persistent>
        <priority>-1</priority>
        <expiryTime>-1</expiryTime>
        <format>CBE</format>
      </wmqAdapter>
    </eventDispatcherAdapter>
  </eventDispatcher>
</eventDispatcherSpecification>
```

Figure 5-62 Event Dispatcher part of event binding file

7. Export the ShoppingMonitorBundle to a System z HFS.

Using the IBM CICS Explorer CICS SM perspective we create a new Bundle definition using the following information:

- Name: SHOPMON
- Bundle Directory: /u/cicsrs2/bundles/ShoppingMonitorBundle

Install the bundle in CICS.

8. Test the application again as described in 5.7, “Testing the Shopping application” on page 119.
9. In addition, test the Order Item, Fulfill Order, and Ship Order functions, as shown below.
- a. Place an order as shown in Figure 5-63 on page 147. Specify Customer Number as 00001 and press PF2.

```
Order application

Customer Number: . . . . . 00001

PF1    Query Item

PF2    Order Item

PF4    fulfill Order

PF5    Ship Order

Exit = PF3
```

Figure 5-63 Test Order Item function

b. On the next panel (Figure 5-64), enter the following information:

- Item Number: 00001
- Quantity: 00001

Press **Enter**.

```
Order Item

Customer Number: . . . . . 00001

Item Number: . . . 00001

Quantity : . . . . 00001

Please enter item number and press Enter

Exit = PF3
```

Figure 5-64 Order an item

The order confirmation panel (Figure 5-65) is shown. Press **PF3** to return to the MENU panel.

Order Confirmation

Customer Number: 00001

Order Number: . . 00044

Item Number: 00001

Quantity: 00001

Total Value of Order: . . .

1.99

PF3 = Menu Screen

PF12 = Exit

Figure 5-65 Order Confirmation panel

- c. On the MENU panel, press **PF4** to fulfill the order and **PF5** to ship the order.

The emitted events are being consumed by WebSphere Business Events. See Chapter 7, “WebSphere Business Events scenario” on page 183 and WebSphere Business Monitor as shown in Chapter 8, “WebSphere Business Monitor scenario” on page 213.



Troubleshooting

In this chapter we discuss best practices, performance considerations, and illustrate the monitoring and statistics that are available for event processing. We provide a step-by-step procedure to determine why expected events are not captured as well as guidance to ascertain why too many events are captured and why the data captured may not be as expected. We show how to check why emitted events are not received by WebSphere Business Events and WebSphere Business Monitor.

6.1 Best practices for performance

From a performance perspective, CICS event processing consists of three main steps:

- ▶ Capturing an event
- ▶ Dispatching the EP adapter
- ▶ Emitting the event through the EP adapter

6.1.1 Capturing an event

Filtering of the predicates specified for each event occurs during the capture stage. The capture point specifies the API call for which we want to capture an event. The filter is made up of predicates that identify the circumstances in which instances of that API call relate to the event. So, when event processing is started and no event bindings are installed, processor usage is negligible. Processor usage is also negligible when event bindings are installed, but there is no capture specification for this API call.

The primary predicate is the primary value specified in the API command. For example, for a **PUT CONTAINER** command, the container name is a primary predicate. On a **READ FILE**, the file name is a primary predicate. Filter performance is optimized by the use of primary predicate filters that use the “Equals” operator. If there is a match on the primary predicate, use of “Equals” can result in improved performance compared to other operator values.

Reduce the number of filters, where possible, by not including unnecessary filters. For example, if only one program writes to file XYZ, it is only necessary to filter on the file name and not the program name as well.

Another way to help with performance is to organize the filter predicates to put those that filter out the largest number of unwanted events before the filter predicates that exclude fewer unwanted events, in the capture specification.

Do not use filter predicates on zoned decimal or packed decimal data fields that might contain unusable data. Ensure you have sufficient filter predicates prior to making a packed decimal or zoned decimal check, to ensure that the data area being checked always contains valid packed decimal or zoned decimal data.

If you capture payload data associated with the event (information sources, in the CICS Explorer event binding editor), keeping the number of data items captured to a minimum aids performance. For example, it is more efficient to define a single data capture element to capture bytes 0–35 than it is to define five data capture elements capturing bytes 0–5, 10–15, 20–25 and 30–35. It is

recommended that containers do not contain a mix of data types. If the event is being emitted to another platform, it is important that character data is translated to the correct code page.

6.1.2 Dispatching the EP adapter

In the dispatching stage, event processing runs multiple dispatcher tasks on L8 TCBs to emit events. The maximum number of L8 TCBs used for event processing dispatchers is limited to one third of the system initialization parameter value for MAXOPENTCBS to prevent event processing from monopolizing the L8 TCBs. You can use the data in the “Peak event capture queue” and “Peak dispatcher tasks” entries in the EVENTPROCESS global statistics (Example 6-23 on page 166) to ensure that your MAXOPENTCBS value is not set too low.

Specifying a user ID or transaction ID in the advanced options of the EP adapter causes the EP adapter to be attached as a separate task. However, performance is improved if no user ID or transaction ID is specified in the EP adapter, as the EP adapter is then linked to from the dispatcher task.

6.1.3 Emitting an event through the EP adapter

During the EP adapter stage, the cost of running the EP adapters depends on which EP adapter is being used. The TSQ EP adapter imposes the least cost in terms of CPU consumed. An MQPUT1 call is generated for each event emitted to the WMQ EP adapter, which has some overhead.

The transaction start EP adapter starts a new CICS task. Therefore, processor usage increases due to starting the transaction that is driven as a result of the CICS event. Also, as each data capture field is made available to a started transaction in a separate container, a large number of data capture fields can impose a significant increase in processor usage on the EP adapter.

6.2 Monitoring and statistics

In this section we look at the monitoring and statistics for events in CICS.

6.2.1 Monitoring

CICS monitoring collects data about the performance of all user and CICS transactions during online processing, for later offline analysis. The records produced by CICS monitoring are MVS System Management Facility (SMF) type 110, and are written to an SMF data set. Monitoring can be set on by various means:

- ▶ CICS system initialization parameter MN
- ▶ CICS master terminal transaction CEMT SET MONITOR ON
- ▶ Transaction CEMN
- ▶ SPI command **EXEC CICS SET MONITOR**

In CICS TS 4.1, the following new performance class data fields containing event processing information are added to the DFHCICS group:

- ▶ EICTOTCT
The total number of **EXEC CICS** commands issued by the user task.
- ▶ ECSIGECT
The number of **EXEC CICS SIGNAL EVENT** commands issued by the user task.
- ▶ ECEFOPCT
The number of event filter operations performed by the user task.
- ▶ ECEVNTCT
The number of events captured by the user task.

The CICS Explorer, with the CICS Performance Analyzer plug-in, can be used to view the event processing monitor data. Figure 6-1 on page 153 contains an example of the performance class data fields for event processing as they appear in CICS Performance Analyzer.

pingEventBinding.evbind HDB2.CSV (3) X									
HDB2.CSVEPRED5+									
me	MVS	Applid	Transact...	Task number	User ID	EICTotCt	ECEFOPCT	ECEVNTCT	ECSIGECT
5.712	SC66	EPRED5	MENU	83	CICSUSER	157	45	6	1
3.365	SC66	EPRED5	CSAC	84	CICSUSER	0	0	0	0
0.279	SC66	EPRED5	CEBR	85	CICSUSER	16	0	0	0

Figure 6-1 CICS Explorer, CICS Performance Analyzer plug-in showing event processing monitor data

The data in Example 6-1 is generated from the records contained in the SMF data sets, using the monitoring dictionary utility program DFHMNDUP and the sample monitoring data print program DFH\$MOLS. This example shows that 1391 EXEC CICS commands were issued, including one SIGNAL EVENT. Due to enabled capture specifications, two filter operations were carried out, resulting in two events being captured.

Example 6-1 Monitor data for Event Processing.

--FIELD-NAME-----			UNINTERPRETED-----	INTERPRETED-
DFHCICS	A402	EICTOTCT	00 00056F	1391
DFHCICS	A415	ECSIGECT	00 000001	1
DFHCICS	A416	ECEFOPCT	00 000002	2
DFHCICS	A417	ECEVNTCT	00 000002	2

We find more detailed information about these events from the statistics utility program DFH0STAT, in the next section.

6.2.2 Statistics

CICS gathers statistics data about the system resource usage and the performance of the CICS system during online processing. The CICS statistics domain creates MVS System Management Facility (SMF) type 110, sub-type 2 records, which are written to an SMF data set. Statistics data is useful both for performance tuning and for capacity planning.

The recording of statistics can be started by the following means:

- ▶ The CICS system initialization parameter STARTRCD=YES, which causes interval statistics to be recorded every three hours, by default.
- ▶ CICS Master terminal transaction CEMT SET STATISTICS(ON).
- ▶ SPI command **EXEC CICS SET STATISTICS RECORDING(ON)**.

The sample statistics utility program (DFH0STAT) is used to generate reports about the event processing statistics. Global statistics can be requested for EVENTPROCESSING and global and resource statistics can be requested for EVENTBINDINGS and CAPTURESPECS.

Install the resource group DFH\$STAT in your CICS region and run the STAT transaction. Once in the STAT application, use PF4 Reports and scroll to the page containing the event processing, EVENTBINDINGS and CAPTURESPECS reports. You can also select the BUNDLES report, which provides details of all of the bundles currently installed. Example 6-2 shows the section of the online STAT application where you can select the event processing statistics.

Example 6-2 Selecting event processing statistics in the STAT application

Sample Program - CICS Statistics

Select the statistics reports required

DB2 Connection	N
DB2 Entries	N
JVM Pool and Class Cache	N
JVMs	N
JVM Profiles	N
JVM Programs	N
JVMSERVERs	N
BUNDLES	N
Event Processing	Y
EVENTBINDINGS	Y
CAPTURESPECS	Y
XMLTRANSFORMs	N

In the generated output, EVENTPROCESS global statistics (Example 6-4 on page 156) include event processing queue status and tasks, as well as the number and type of events captured. Example 6-2 shows that event processing is started, so events are being captured for event bindings that are installed and enabled. In this instance, three events have been generated and all are non-transactional (because the number of transactional events is zero) and normal priority events. All three events were emitted to the TSQ EP adapter.

“Events lost (dispatcher) - config” events are events that were captured but not dispatched to an EP adapter because the dispatcher encountered a problem relating to information specified in the EP adapter advanced options (for example, if an invalid user ID or transaction is specified).

“Events lost (dispatcher) - other” events are events that were captured but not dispatched to an EP adapter because the dispatcher encountered a problem in the CICS environment (for example, insufficient storage).

Example 6-3 Event Processing global statistics

```

Event Processing
Event Processing Status . . . . . : Started
Put Events. . . . . : 3
Sync commit events. . . . . : 0
Sync backout events . . . . . : 0
Current event capture queue . . . . : 0
Peak event capture queue. . . . . : 1
Current transactional queue . . . . : 0
Peak transactional queue. . . . . : 0
Normal events . . . . . : 3
Priority events . . . . . : 0
Transactional events. . . . . : 0
Transactional events discarded. . . : 0
Dispatcher tasks attached . . . . . : 3
Current dispatcher tasks. . . . . : 2
Peak dispatcher tasks . . . . . : 3
Events to WMQ EP adapter. . . . . : 0
Events to Transaction EP adapter. . : 0
Events to Tsqueue EP adapter. . . . : 3
Events to Custom EP adapter . . . . : 0
Events lost (dispatcher) - config . : 0
Events lost (dispatcher) - other. . : 0

```

EVENTBINDING global statistics under the Event Capture heading in the statistics report (Example 6-4 on page 156) include the number of filter operations, events generated for event bindings that have since been disabled, and total events captured for each event binding.

In this report, “Filter operations failed” events are events that may have been lost because filtering was attempted but failed, so there might not have been a match that was missing.

“Capture operations failed” events are events that have been lost.

“Events lost (adapter) - config” events are events that were captured but not emitted because the EP adapter encountered a problem relating to information specified in the EP adapter section of the event binding. For example, events appear here if an incorrect queue name is specified when using the WMQ EP adapter.

“Events lost (adapter) - other” events are events that were captured but not emitted because the EP adapter encountered a problem in the CICS environment, such as, for example, insufficient storage.

“Events lost (adapter)” information is available for the CICS-provided EP adapters, but not for problems with custom EP adapters.

Example 6-4 Event Binding Global Statistics

Event Capture	
Event Filtering operations	72
Events with disabled EVENTBINDING . .	0
Events captured	3
Filter operations failed.	0
Capture operations failed	0
Events lost (adapter) - config. . . .	0
Events lost (adapter) - other	0

EVENTBINDING resource statistics include a list of individual event binding names and their enable status (Example 6-5).

Example 6-5 Event Binding Resource Statistics

EVENTBINDINGS	
EVENTBINDING Name	ShoppingEventBinding
Enable Status	Enabled

Additional information is available for the event binding resource using the SPI command **EXTRACT STATISTICS**. As well as the event binding name and status, this SPI command returns the resource change statistics, which are mapped by the copy book DFHECRDS.

The CAPTURESPEC resource statistics report shows statistics about each event capture specification, including the capture specification name, its associated event binding, the enable state of the event binding, type and name of the capture point, the business event name, and the total events captured by this capture specification. In this report we can see which capture specifications within our event binding have successfully captured events (Example 6-6 on page 157).

CAPTURESPECs

EVENTBINDING Name : ShoppingEventBinding

Enable Status : Enabled

Capturespec name	Capture point type	Capture point	Events Captured	Event name
CaptureFulfil	POSTCOMMAND	PUT_CONTAINER	0	Fulfil
CaptureOrder	POSTCOMMAND	LINK_PROGRAM	1	Order
CaptureQueryEvent	PROGRAMINIT	PROGRAM_INITIATION	0	Query
CaptureShip	POSTCOMMAND	REWRITE	1	Ship
CaptureSendOrder	POSTCOMMAND	SIGNAL_EVENT	1	SendOrder

6.3 Problem determination

There may be times when events that are expected to be received are not captured, or when you capture more events than intended. There are several tools that can be used to determine why the expected number of events is different from the actual number captured:

- ▶ Statistics utility program DFH0STAT
- ▶ CICS auxiliary trace
- ▶ CICS dump
- ▶ CICS Explorer
- ▶ Master terminal transaction CEMT INQUIRE
- ▶ CICSplex SM WUI

In this section we demonstrate how to use these tools when expected events are not captured by event processing. We discuss some reasons why unexpected events are captured and how to determine why captured data is missing, incorrect, or contains only asterisks. We also provide information about taking steps to find the cause of missing events that have been captured but have not arrived at their destination, if the events are to be emitted through a WMQ EP adapter to either WebSphere Business Events or WebSphere Business Monitor.

In addition to the above list of tools, you can use the TSQ EP adapter and the sample custom EP adapter during problem determination check the number of events and the data being captured. The TSQ EP adapter writes CICS event objects to a TS queue in CICS flattened event format, and the sample custom EP adapter writes them to a TS queue in a simplified flattened event format.

6.3.1 Events not captured

This section illustrates techniques to determine why expected events are not captured when an order process is completed in our shopping sample application.

The event binding used in this section (ShoppingEventBinding) contains five events and each has one capture specification as shown in 3.4, “The events emitted in the scenarios” on page 52.

To generate events, run the sample application to order an item, fulfill the order, and ship the order. We expect one event to be captured for each of these tasks, as well as one SIGNAL EVENT API call to be captured, for a total of four events.

Our event binding is created to emit events to the TSQ EP adapter. In this instance, we run the sample application and find that no events are captured at all. The first check to make is that CICS is capable of capturing events. Use the CICS Explorer Event Processing view to check the EVENTPROCESS status is set to STARTED. If EVENTPROCESS is STOPPED, no events will be captured. In our case, EVENTPROCESS is started.

Next, check that the event binding is installed and enabled. This can be done using the CICS Explorer. See Chapter 5, “Generating events” on page 91. The master terminal commands **CEMT INQUIRE EVENTPROCESS** and **CEMT INQUIRE EVENTBINDING**, and the CICSplex SM WUI can also be used to gather this information.

Using the CICS Explorer, we find that our event binding is not installed and that the bundle is installed but disabled. This is an indication that something could be wrong with the event binding. There is a group of messages written to the CICS message log each time a BUNDLE resource is installed.

The first message to search for is DFHRL0107, indicating that the CICS resource life cycle manager has started to create the BUNDLE resource. If the BUNDLE resource has failed to be created, or if an event binding has failed to be installed successfully, there will be additional messages indicating the cause of the failure. For example, in this instance message, DFHPI1007 is seen in the CICS message log indicating a problem with the XML data in the event binding (Example 6-2 on page 154).


```
DFHRL0107 I 07/10/2009 05:06:11 EPRED5 CICSUSER The CICS reesource
  life-cycle manager has started to create the BUNDLE resource
  SHOPPING.
DFHPI1007 07/10/2009 05:06:11 EPRED5 00070 XML to data
  transformation failed because of incorrect input
  (XML_FORMAT_ERROR Content data found outside root element) for
  EVENTBINDING ShoppingEventBinding.
```

Figure 6-2 Message DFHRL0107 followed by message DFHPI1007, which indicates a problem with the event binding

The next message to look for in the CICS message log is a message with the DFHEC prefix. In our example, DFHEC1003 (Figure 6-3) appears. This message is issued for several different reasons, as indicated in the reason at the end of the message. In this instance, the reason indicates BAD XML DATA. Message DFHRL0102 confirms that the event binding resource has not been created.

```
DFHEC1003 07/10/2009 05:06:11 EPRED5 The CICS event capture component
  failed to create the EVENTBINDING resource ShoppingEventBinding for
  reason BAD_XML_DATA.
DFHRL0102 E 07/10/2009 05:06:11 EPRED5 CEDA The CICS resource life-cycle
  manager failed to create the resource ShoppingEventBinding and
  returned with reason CALL BACK ERROR.
```

Figure 6-3 Messages showing an event binding has failed to be created in CICS

It is important at this stage to examine the event binding to establish the cause of the bad XML. One cause of bad XML data is if the event binding is manually transferred from the CICS Explorer workspace on the local workstation to the HFS using file transfer protocol (FTP) without specifying binary transfer. Browse the event binding file in the HFS and check that the file is in ASCII format. If not, FTP the file again from the workstation or deploy the bundle from the CICS Explorer. See Chapter 5, “Generating events” on page 91.

When an event binding is successfully installed, the DFHRL0107 message will be followed by DFHEC1001, as in Figure 6-4 on page 160, which confirms the event binding has been created successfully. Message DFHRL0109 confirms the BUNDLE resource is installed and enabled.

```

DFHRL0107 I 07/10/2009 05:07:46 EPRED5 CICSUSER The CICS resource life-
cycle manager has started to create the BUNDLE resource SHOPPING.
DFHEC1001 07/10/2009 05:07:46 EPRED5 Event binding ShoppingEventBinding
installed successfully.
DFHRL0109 I 07/10/2009 05:07:46 EPRED5 CEDA The CICS resource life-cycle
manager has created the BUNDLE resource SHOPPING and the BUNDLE is in
the enabled state.

```

Figure 6-4 Messages showing a bundle and event binding have been installed successfully in CICS

Once the event binding is created and enabled, run the shopping sample application again to order an item, fulfill the order, and ship the order. As before, we expect four events to be captured. However, only three events appear on our TS queue. Therefore, this section contains a sequence of steps to take to determine why we are not receiving the correct number of events.

The event processing global statistics reports can give us more details about the number of events being generated. In Example 6-3 on page 155, “Put events” shows the total number of captured events passed to the EP adapter, through the dispatcher. “Normal events” shows the total number of normal priority events generated. These lines are shown in Example 6-7.

Example 6-7 Extract from event processing global statistics

Put Events.	:	3
Normal events	:	3

Our event binding has been configured with a TSQ EP adapter, so the line from the event processing global statistics is also relevant (Example 6-8).

Example 6-8 TSQ EP adapter data in event processing global statistics

Events to Tsqueue EP adapter. . . .	:	3
-------------------------------------	---	---

However, as we were expecting four events but only three are shown in the statistics reports, we need to determine why one event has not been generated.

In the first instance, view the event binding in the CICS Explorer event binding editor, to determine if there is an obvious reason why the expected number of events are not being captured by a particular capture specification. Check the filter predicates for errors that might lead to events not being captured. If the cause of the missing events cannot be determined from the event binding,

generate the CAPTURESPEC resource statistics report using the DFH0STAT sample statistics application. See 6.2.2, “Statistics” on page 153.

CAPTURESPEC resource statistics lists details about each capture specification. Example 6-9 shows that we received events for the Order, Ship, and SendOrder events, but not for the Fulfill event (we do not expect a QueryStock event, as we have not used this part of the shopping sample application).

Example 6-9 Capture specification resource statistics

CAPTURESPECs

EVENTBINDING Name : ShoppingEventBinding

Enable Status : Enabled

Capturespec name	Capture point type	Capture point	Events Captured	Event name
CaptureFulfil	POSTCOMMAND	PUT_CONTAINER	0	Fulfil
CaptureOrder	POSTCOMMAND	LINK_PROGRAM	1	Order
CaptureQueryEvent	PROGRAMINIT	PROGRAM_INITIATION	0	Query
CaptureShip	POSTCOMMAND	REWRITE	1	Ship
CaptureSendOrder	POSTCOMMAND	SIGNAL_EVENT	1	SendOrder

The capture specification for the Fulfill event is called Capturefulfil, and is designed to capture an event on a PUT CONTAINER API call in the program called UPDSTOCK, when the container name equals OUTPUT and when the stock level falls to less than 50.

So, we generate and format a level 1 and 2 CICS auxiliary trace for the EC component, to determine why the Fulfill event was not captured.

Within the trace, we search for the group of ECEC entries that could relate to the following PUT CONTAINER API command:

```
EXEC CICS PUT_CONTAINER('OUTPUT')
FROM(UPDATE-OUT)
END-EXEC.
```

We find the entries showing the event for the PUT_CONTAINER predicate and the container name is OUTPUT, as shown in Example 6-10 on page 162. “POST-CMD” indicates the event is captured after the command completes.

Example 6-10 CICS trace entry showing the PUT_CONTAINER predicate

```
AP 3530 ECEC ENTRY - FUNCTION(EVENT_CAPTURE) PARM_LIST(27049BD8) EVP_ADDRESS(260CFE31)
:POST-CMD(PUT_CONTAINER)
      AB000000 00000000 01000100 27049BD8 *.....,.....Q*
      *.....*
      00000000 000DD7E4 E36DC3D6 D5E3C1C9 *.....PUT_CONTAI*
      09C3D6D5 E3C1C9D5 C5D90000 00 *NER.....CONTAINER... *
```

```
DD 0301 DDLO ENTRY - FUNCTION(LOCATE) DIRECTORY_TOKEN(26E1CF20) ENTRY_NAME(26E2F197)
LOCKING MODE(CALLER) DIRECTORY NAME(ECCS) NAME(....OUTPUT.....)
```

Continuing our search through the trace, we find that the next ECEC trace entry confirms that the primary predicate has matched. The container name is OUTPUT, as in Example 6-11.

Example 6-11 CICS trace entry showing our primary predicate is matched

```
AP 353C ECEC EVENT - PRIMARY_PREDICATE MATCHED
      40404040 00000000 00000000 00000000 *....OUTPUT .....
```

The next ECEC trace entry, shown in Example 6-12, is for our capture specification, called Capturefulfill.

Example 6-12 CICS trace entry for our capture specification

```
AP 353B ECEC EVENT - FILTERING CAPTURESPEC(CaptureFulfil)
      C38197A3 A49985C6 A4938689 93404040 *.y>DFHECCS CaptureFulfil *
```

In this trace entry we see that the current program name is equal to 'UPDSTOCK', exactly as has been selected in our event binding (Example 6-13).

Example 6-13 CICS trace entry showing our CURRENT_PGM predicate is true

```
AP 3534 ECEC EVENT - PREDICATE_TRUE - TESTING CURRENT-PROGRAM EQUAL
      27289350 00040000 00000000 00000000 *...>DFHECFP ..l&.....*
      C5D5E36D D7C7D440 *.....CURRENT_PGM *
      *UPDSTOCK *
      *UPDSTOCK *
```

However, the following ECEC trace entry (Example 6-14) gives an indication as to why the event is not captured. The test comparing the new level of stock (in the OUTPUT container) and our predicate of the filter value is found to be false. The values in this trace entry show that the current level of stock is 45, whereas our predicate is “less than 5”.

Example 6-14 CICS trace entry showing our LESS_THAN predicate is false

```

AP 3539 ECEC EVENT - PREDICATE_FALSE - TESTING ZONED IN PARAMETER (FROM) LESS_THAN
      00000000 001E0802 40050080 10021005 *..>DFHECFP .....*
      40404040 40404040 *.....FROM .....*
                                   *00045 .....*
                                   *00005 .....*

```

The final trace entry confirms that this event has not been captured (Example 6-15).

Example 6-15 CICS trace entry showing this event has not been captured

```

AP 3531 ECEC EXIT - FUNCTION(EVENT_CAPTURE) RESPONSE(OK) EVENTS(0)

```

To further confirm the findings within the trace, we use a CICS dump formatted for the EC component. In our CICS region we generate a system dump using the **CEMT PER SNAP** command.

Format the system dump using Interactive Problem Control Facility (IPCS) on z/OS, with the **VERBX CICS660 'EC=3'** command.

We search in the formatted dump for the sub-title ‘Event Binding Summary’. The Event Binding Summary section lists details about each event binding installed in our CICS system, along with all of the capture specifications, filter predicates and the number of events captured for each event binding.

The important pieces in the summary for our event binding can be seen in the following examples (Example 6-16 on page 164 to Example 6-18 on page 164). The first line in the ECEVB control block contains our event binding name, its enable status, and the EP adapter type we are using (Example 6-16 on page 164).

Example 6-16 The start of the ECEVB control block

```
EC: EVENT BINDING SUMMARY
ECEVB: ShoppingEventBinding      UserTag=V001      UserId=..... TranId=....
AdapterType=TSQ Enabled=Y Transactional=N Durable=N UseContextUserId=N
AdapterAttached=N Priority=NORMAL
```

The ECCS: entry shows this is our capture specification called Capturefulfill, along with the primary predicate of PUT_CONTAINER, for our container named OUTPUT (Example 6-17).

Example 6-17 Capture specification information in the ECEVB control block

```
ECCS: CaptureFulfil PUT_CONTAINER(postcmd)group/function= 3416 Events=0000000000000000
Specific Primary_Predicate: CONTAINER='OUTPUT'
Filter type Keyword Op Offset Length ...KB KM MN Value(up to 32chars)
ECFP: program CURRENT_PGM EQ 00000000 00000008 ...00 00 00 'UPDSTOCK'
ECFP: fulldata_zoned FROM LT 0000000D 00000005 ...05 00 10 '00005'
```

The ECFP: entries contain the filter predicates: CURRENT_PGM equals UPDSTOCK and data item at offset 13 (x'D') in the OUTPUT container to be less than '00005' (Example 6-18).

Example 6-18 Filter predicate information in the ECEVB control block

```
Filter type Keyword Op Offset Length ...KB KM MN Value(up to 32chars)
ECFP: program CURRENT_PGM EQ 00000000 00000008 ...00 00 00 'UPDSTOCK'
ECFP: fulldata_zoned FROM LT 0000000D 00000005 ...05 00 10 '00005'
```

So, we had thought that our capture specification was designed to check for a stock level of less than 50, but the trace entries and the event binding summary in the formatted dump show that we have actually specified a predicate of “stock level of less than 5” in the event binding. This is corrected by changing the application data predicate in the filtering view for the event binding in the CICS Explorer event binding editor and saving the event binding. Re-deploy the amended event binding to the HFS and re-install the bundle into CICS, to pick up the changes made to the event binding.

6.3.2 Unexpected events captured

To determine why more events are being captured than expected, we start by identifying which capture specification is capturing the unexpected events, using the statistics utility program DFH0STAT. Install the group DFH\$STAT and run the STAT application. Press PF4 Reports and scroll to the page containing the CAPTURESPECS report, select it, and press Enter to generate the report.

The CAPTURESPECS report (Example 6-19) lists the capture specifications for our event bindings and the number of events captured for each capture specification. So, if we have more than an expected number of events generated, we can quickly identify which capture specification is generating the extra events.

Example 6-19 The CAPTURESPECS statistics report

CAPTURESPECS				
EVENTBINDING Name		ShoppingEventBinding		
Enable Status		Enabled		
	Capture		Events	
Capturespec name	point type	Capture point	Captured	Event name
<hr/>				
CaptureFulfil	POSTCOMMAND	PUT_CONTAINER	1	Fulfil
CaptureOrder	POSTCOMMAND	LINK_PROGRAM	1	Order
CaptureQueryEvent	PROGRAMINIT	PROGRAM_INITIATION	0	Query
CaptureShip	POSTCOMMAND	REWRITE	1	Ship
CaptureSendOrder	POSTCOMMAND	SIGNAL_EVENT	1	SendOrder

For this example we use the capture specification called Capturefulfill. To check why the event was generated for the Capturefulfill capture specification, we generate and format a CICS auxiliary trace, containing level 1 and 2 trace entries for the EC component. Using the information from the CAPTURESPEC statistics, we search for the ECEC trace entries relating to our capture specification named Capturefulfill (Example 6-20).

Example 6-20 CICS trace entry showing our capture specification name

```
AP 353B ECEC EVENT - FILTERING CAPTURESPEC(CaptureFulfil)
      C38197A3 A49985C6 A4938689 93404040 *.y>DFHECCS      CaptureFulfil *
```

Just prior to this entry in the trace, we find the ECEC EVENT entry with PREDICATE_TRUE, as in Example 6-21 on page 166. The predicate of “Less than 50” for the new stock level is found to be true, as the actual stock level is 45, so the event is captured.

Example 6-21 CICS trace entry showing our LESS_THAN predicate is true

```
AP 3534 ECEC EVENT - PREDICATE_TRUE - TESTING ZONED IN PARAMETER(FROM) LESS_THAN
00000000 001E0802 40050080 10021005 *..>DFHECFP .....*
40404040 40404040 *.....FROM *
*00045 *
*00050 *
```

The two previous examples (Example 6-20 on page 165 and Example 6-21) illustrate why the event was captured for this capture specification. Using this technique can help you determine why unexpected events are captured. In the examples used here, the events were expected.

If too many events are being emitted to an EP adapter, use the EVENTPROCESS global statistics to identify the number of events destined for each EP adapter type (Example 6-22). For more information, see 6.2.2, “Statistics” on page 153.

Example 6-22 Extract from Event Processing Global Statistics showing the number of events emitted to each EP adapter type

Events to WMQ EP adapter. :	0
Events to Transaction EP adapter. . :	0
Events to Tsqueue EP adapter. . . . :	3
Events to Custom EP adapter :	0

You can also use a CICS dump to check the number of events sent to each EP adapter. Format the dump for the EC component and search for the subtitle “Event Binding Summary”. The event binding summary contains the ECEVB control block (as shown in Example 6-16 on page 164 to Example 6-19 on page 165), and lists the event bindings, the capture specifications, and the events captured. The ECEVB also contains the EP adapter type, which can be used to identify the events sent to each EP adapter (Example 6-23).

Example 6-23 The ECEVB control block showing the EP adapter type for this event binding

```
EC: EVENT BINDING SUMMARY
ECEVB: ShoppingEventBinding   UserTag=V001   UserId=..... TranId=....
AdapterType=TSQ
```

6.3.3 Capture data is not as expected

When capturing an event you can supply a payload for the event, which can include application context data, command options, and items of application data. If the data that is captured is not the data that was expected, we present here some approaches you may use to identify the cause.

If some or all of your application data is missing from the expected payload, this could be due to an optional parameter, channel, or container not being present at the time the event is captured. It could also be due to a container, data area, or COMMAREA having been provided that is too short to hold the data item.

If all of the expected data is missing, gather a CICS auxiliary trace, with level 1 and 2 data for the EC component. Format the trace, and search for the phrase “UNAVAILABLE_DATA” within an ECEC trace record. The trace record will contain the source of the data and ECEC trace records just prior to this will identify the capture specification for this event. Using this information, you can check the circumstances under which you are expecting to capture data. For example, there may be occasions when the data is genuinely not available. In this instance you can adjust the predicate information in the event binding to allow for this.

If data has been captured with the event, but it is not the correct data, this may have been caused by incorrect filtering information in the capture specification. This can be verified by viewing the information sources for the event binding, in the CICS Explorer event binding editor. Check that the correct data types, lengths, offsets, and sources have been used for your emitted business information. For example, you may have fields of different formats, so check that all the data formats are correct in the capture specification. A CICS auxiliary trace with level 1 and 2 entries for the EC component can help identify these fields.

Capturing data from the wrong parameter or container, or specifying the wrong offset, length, or both in the capture specification could also lead to incorrect data being seen in the event. In this instance, a CICS dump formatted for the EC component is useful, Capture and format the dump as described on page 163.

Search the Event Binding Summary in the dump, and find the event binding and capture specification that is generating the incorrect data. The ECCD: entry (or entries, if there is more than one item of data being captured) for this capture specification contains the data being captured, its length and where it is being captured from. Use this to identify any incorrect information in the capture specification of the event binding.

If the captured data is numeric and is too large for the formatted field length, then the entire field is replaced with asterisks. Check the length specified in the capture specification and adjust as necessary. The event data will also contain asterisks if the data requested to be captured is missing entirely.

6.3.4 Captured events are missing from WebSphere Business Events

This section describes how to diagnose problems when the event binding is configured to emit events, through a WMQ EP adapter, to WebSphere Business Events. In our environment, we use WebSphere Business Events running on z/OS, and show you how to diagnose problems in that configuration, but there are equivalent ways to check DB2® and WebSphere MQ on other platforms.

To determine the cause of missing events, we start with the basics and use the CICS Explorer to check that the EVENTPROCESS status in our CICS region is set to STARTED. Once this is confirmed, use the CICS Explorer to check that the event binding is installed and enabled. See Chapter 4, “Setting up CICS for events” on page 57

You could also use the master terminal commands **CEMT INQUIRE EVENTPROCESS** and **CEMT INQUIRE EVENTBINDING**, or the CICSplex SM WUI to gather this information.

Check that events are being dispatched to the WMQ EP adapter, for example, by generating statistics using the CICS DFH0STAT sample program (as in 6.2.2, “Statistics” on page 153).

Once it is verified that the events have been captured, there are further steps that can be taken to determine why they cannot be seen in the event report function in the WebSphere Business Events Administration console.

To begin troubleshooting, stop the WebSphere Business Events connector job that is running to confirm that the events are arriving on the WebSphere Message Queue and are not removed from the WebSphere Message Queue in the meantime.

Ensure DB2 on z/OS is running with the **/-D9E 1 DISPLAY GROUP** command, confirmed by the output shown in Example 6-24 on page 169.

Example 6-24 Output displaying the status of DB2 on z/OS.

```

DSN7100I  -D9E1 DSN7GCMD
*** BEGIN DISPLAY OF GROUP(DB9EU  ) GROUP LEVEL(910) MODE(N )
          PROTOCOL LEVEL(2)  GROUP ATTACH NAME(D9EG)
-----
DB2                DB2 SYSTEM      IRLM
MEMBER  ID  SUBSYS  CMDPREF  STATUS  LVL NAME      SUBSYS  IRLMPROC
-----
D9E1      1  D9E1   -D9E1   ACTIVE   910 SC66      I9E1   D9E1IRLM
D9E2      2  D9E2   -D9E2   ACTIVE   910 SC53      I9E2   D9E2IRLM
-----
SCA  STRUCTURE SIZE:    10240 KB, STATUS= AC,   SCA IN USE:    2 %
LOCK1 STRUCTURE SIZE:    9216 KB
NUMBER LOCK ENTRIES:    2097152
NUMBER LIST ENTRIES:    11111, LIST ENTRIES IN USE:    9
*** END DISPLAY OF GROUP(DB9EU  )
DSN9022I  -D9E1 DSN7GCMD 'DISPLAY GROUP ' NORMAL COMPLETION

```

Start the WebSphere MQ queue manager on z/OS and the WebSphere MQ channel initiator.

Start the WebSphere Application Server on z/OS that hosts WebSphere Business Events and wait for the message BBOO00191 to appear in the console log, indicating the server has started. Switch to the Display Active panel in SDSF and confirm the WebSphere Application Server job (with prefix WQ6*) is active, as shown in Example 6-25.

Example 6-25 SDSF Display Active panel showing the active WebSphere Application Server jobs on z/OS.

```

SDSF DA SC66 (ALL)    PAG    0 CPU/L/Z    5/  3/  0    LINE 1-4 (4)
COMMAND INPUT ==>
PREFIX=WQ6*  DEST=(ALL)  OWNER=**  SORT=JOBNAME/A  SYSNAME=*
NP  JOBNAME  StepName  ProcStep  JobID    Owner    C Pos DP Real
WQ6661  WQ6661  BBOCTL  STC12521  WQACRU    NS  F4  59T  0.00
WQ6661A  WQ6661A  BBOSR  STC12525  WQASRU    IN  F2  90T  0.00
WQ6661D  WQ6661D  BBODAEMN  STC12524  WQACRU    NS  FE  6937  0.00
WQ6661S  WQ6661S  BBOSR  STC12526  WQASRU    IN  F2  167T  0.00

```

Submit the WebSphere Business Events connector job and check that its job output STDOUT contains the messages that show that the appropriate events/actions are being monitored, as shown in Example 6-26.

Example 6-26 Messages in the WebSphere Business Events job output.

```
BEER0642I: The file system module is monitoring action: Send discount offer
BEER0642I: The file system module is monitoring action: Order notification
BEER0642I: The file system module is monitoring action: Notify Query
BEER0616I: The JMS module is monitoring event: Query
```

Check that the WebSphere MQ queue being used for events has an open input count of one and the application accessing it is the connector job (WQ6661C in our case), as shown in Example 6-27.

Example 6-27 Panel showing status of the WebSphere MQ queue that is being used for events.

List Open Queues - MQ8G						Row 1 of 2	
Queue name		Disposition		Access			
Application		ASID	Application information	User ID	State		
External URID		UR type		MQ URID			
<>	PJB*			QMGR	MQ8G		
	PJB_WBE_QUEUE			QMGR	MQ8G	- IX B Q	
	WQ6661C5 BATCH	009F		CICSR56		ACTIVE	
***** End of list *****							

6.3.5 Captured events are missing from WebSphere Business Monitor

This section describes how to diagnose problems when the event binding is configured to emit events, through a WMQ EP adapter, to WebSphere Business Monitor.

To start, we return to basics and use the CICS Explorer to confirm that the EVENTPROCESS status in the CICS region is set to STARTED. and to check that the event binding is installed and enabled. See 4.5, “Enabling and disabling and discarding events” on page 74.

Once it has been verified that the events have been captured in CICS (for example, by generating statistics using the CICS DFH0STAT sample program, as in 6.2.2, “Statistics” on page 153), the following steps can be taken to determine why they are not received in WebSphere Business Monitor.

As outlined in 8.3, “WMQ set up, WMQ on z/OS through SIB to CEI” on page 214 the following WebSphere MQ objects (on our queue manager, MQ8G) are defined to enable transportation of the events to the Monitor Server:

- ▶ MQ_TO_CEI (Sender channel)
- ▶ CEI.XMITQUEUE (Transmission queue)
- ▶ WBM.QUEUE (Remote queue definition).

Note: For those not familiar with WebSphere MQ, the event messages will actually be placed on the transmission queue, to enable the sender channel to transmit them to the final destination on the CEI bus on the Monitor Server.

To ensure that these messages have successfully been transmitted, perform the following steps:

Note: If you are unfamiliar with WebSphere MQ commands, check with your MQ Administrator.

1. Check whether there are messages on the transmission queue.

In the Figure 6-5 on page 172, we see that there are three messages in the queue. If the channel is running normally, this should not normally be seen.

```
Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

More:      +

Queue name . . . . . CEI.XMITQUEUE
Disposition . . . . . : QMGR      MQ8G
Description . . . . . : Transmission Queue to SIBus

Put enabled . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . : N  Y=Yes, N=No
Usage . . . . . : X  N=Normal, X=XmitQ
Storage class . . . . . : DEFAULT
CF structure name . . . . . :
Dynamic queue type . . . . . : N  N=Non-dynamic (Predefined), T=Temporary,
                                P=Permanent, S=Shared

Page set identifier . . . . . : 4
Use counts - Output . . . . . : 0          Input . . . . . : 0
Current queue depth . . . . . : 3

Command ==>
```

Figure 6-5 Check transmission queue

2. If there are messages on the transmission queue, check the status of the sender channel. See Figure 6-6 on page 173.

```

List Channels - MQ8G                                Row 1 of 17

Type action codes, then press Enter. Press F11 to display connection
status.
1=Display  2=Define like  3=Alter  4=Manage  5=Perform
6=Start    7=Stop

    Name                                Type              Disposition      Status
<>  *                                CHANNEL          PRIVATE MQ8G
    AJGBRK1.TO.MQ8G                   RECEIVER         QMGR    MQ8G  INACTIVE
    CEI_TO_MQ                         RECEIVER         QMGR    MQ8G  INACTIVE
    MQ_TO_CEI                         SENDER          QMGR    MQ8G  STOP
    MQ8G                               SVRCONN         QMGR    MQ8G  INACTIVE
    MQ8G.TO.AJGBRK1                   SERVER          QMGR    MQ8G  INACTIVE
    MQ8G_TO_QMCIF                     SENDER          QMGR    MQ8G  STOP
    QMCIF_TO_MQ8G                     RECEIVER         QMGR    MQ8G  INACTIVE
    SYSTEM.ADMIN.SVRCONN               SVRCONN         QMGR    MQ8G  INACTIVE
    SYSTEM.BKR.CONFIG                 SVRCONN         QMGR    MQ8G  INACTIVE
    SYSTEM.DEF.CLNTCONN               CLNTCONN        QMGR    MQ8G
    SYSTEM.DEF.CLUSRCVR               CLUSRCVR        QMGR    MQ8G  INACTIVE
    SYSTEM.DEF.CLUSSDR                CLUSSDR         QMGR    MQ8G  INACTIVE
    SYSTEM.DEF.RECEIVER               RECEIVER         QMGR    MQ8G  INACTIVE
    SYSTEM.DEF.REQUESTER              REQUESTER        QMGR    MQ8G  INACTIVE

Command ==>>

```

Figure 6-6 List channels

- If the channel is not running, start the channel. See Figure 6-7 and Figure 6-8 on page 175.

List Channels - MQ8G

Row 1 of 17

Type action codes, then press Enter. Press F11 to display connection status.

1=Display 2=Define like 3=Alter 4=Manage 5=Perform

6=Start 7=Stop

	Name	Type	Disposition	Status
<>	*	CHANNEL	PRIVATE MQ8G	
	AJGBRK1.TO.MQ8G	RECEIVER	QMGR MQ8G	INACTIVE
	CEI_TO_MQ	RECEIVER	QMGR MQ8G	INACTIVE
6	MQ_TO_CEI	SENDER	QMGR MQ8G	STOP
	MQ8G	SVRCONN	QMGR MQ8G	INACTIVE
	MQ8G.TO.AJGBRK1	SERVER	QMGR MQ8G	INACTIVE
	MQ8G_TO_QMCIF	SENDER	QMGR MQ8G	STOP
	QMCIF_TO_MQ8G	RECEIVER	QMGR MQ8G	INACTIVE
	SYSTEM.ADMIN.SVRCONN	SVRCONN	QMGR MQ8G	INACTIVE
	SYSTEM.BKR.CONFIG	SVRCONN	QMGR MQ8G	INACTIVE
	SYSTEM.DEF.CLNTCONN	CLNTCONN	QMGR MQ8G	
	SYSTEM.DEF.CLUSRCVR	CLUSRCVR	QMGR MQ8G	INACTIVE
	SYSTEM.DEF.CLUSSDR	CLUSSDR	QMGR MQ8G	INACTIVE
	SYSTEM.DEF.RECEIVER	RECEIVER	QMGR MQ8G	INACTIVE
	SYSTEM.DEF.REQUESTER	REQUESTER	QMGR MQ8G	INACTIVE

Command ==>

Figure 6-7 Start channel


```
Start a Channel

Select disposition, then press Enter to start channel.

Channel name . . . . . : MQ_TO_CEI
Channel type . . . . . : SENDER
Description . . . . . : Sender channel to SIBus

Disposition . . . . . P      P=Private on MQ8G
                               S=Shared on MQ8G
                               A=Shared on any queue manager

EeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeN
e CSQ9022I -MQ8G CSQXCRPS ' START CHANNEL' NORMAL COMPLETION e
DssssssssssssssssssssssssssssssssssssssssssssssssssssssssM
Command ==>
```

Figure 6-8 Channel started

4. Check that the messages have arrived at the Monitor server.

Note: The sender channel to the CEI bus, by default from the script that created it, is triggered. If you find that the channel is constantly in a stopped state and events are on the transmission queue, ask your MQ administrator to check why the channel initiator triggering is not behaving as expected.

You may not have a monitor model deployed yet to consume these events, or there may be problems with the monitor model (we address this in 8.8, “Troubleshooting WebSphere Business Monitor” on page 296). If this is the case, there is another way for you to verify that the events have arrived at the Monitor server.

There is an application called the Common Base Event (CBE) Browser running on the monitor server that will allow you to view any CBEs received by the server.

Important: To use this application, you need to be persisting the CBEs. This is independent of monitoring, and you may find that this event persistence is not enabled in a runtime environment.

1. To check whether your events are arriving at the CEI, log on to the CBE Browser on the Monitor server:
`http(s)://<hostname>:<admin port>/ibm/console/cbebrowser/events/`
2. Enter a start and end date and time for the event selection. See Figure 6-9.

Event Filter Properties (Optional)

From Creation Date 2009-07-23 To Creation Date 2009-07-23

From Creation Time 09:23:00 To Creation Time 09:25:00

Server Name

Sub-Component Name

From Priority To Priority

From Severity To Severity

SessionID

Get Events

Figure 6-9 Enter selection

3. Click **Get Events**.
4. On the left side of the panel, you will see the number of events returned by the search.
5. As shown in Figure 6-10 on page 177, we see that three events have been received, which corresponds to the three event messages which were stuck on the queue.



Figure 6-10 Returned events

6. Click **All Events** to display the event list.
7. As shown in Figure 6-11, we see in the list that we have received the following information:
 - a. Two events from CICS through the custom EP adapter
These are the events that show com.ibm.wbmonitor.EventEmitter as the Server).
 - b. One event from CICS through the standard event processing
This event shows as CICS EP as the Sub-component).

The image shows a web interface with a table of events. The table has columns for 'Select', 'Creation Time', 'Name', 'Server', and 'Sub-component'. There are three rows of data. The first two rows show events from 'com.ibm.wbmonitor.EventEmitter' with server 'VBMonSrv_wps_CellVBMonSrv_wps_Node\server1' and sub-component 'APT'. The third row shows an event from 'USIBMSC.EPRED4' with sub-component 'CICS EP'. The table is paginated, showing 'Page 1 of 1' and 'Total: 3 Filtered: 3 Displayed: 3 Selected: 0'.

Select	Creation Time	Name	Server	Sub-component
<input type="radio"/>	2009-07-23T09:23:27.687Z	com.ibm.wbmonitor.EventEmitter	VBMonSrv_wps_CellVBMonSrv_wps_Node\server1	APT
<input type="radio"/>	2009-07-23T09:23:27.687Z	com.ibm.wbmonitor.EventEmitter	VBMonSrv_wps_CellVBMonSrv_wps_Node\server1	APT
<input type="radio"/>	2009-07-23T09:24:53.553+00:00		USIBMSC.EPRED4	CICS EP

Page 1 of 1 Total: 3 Filtered: 3 Displayed: 3 Selected: 0

Figure 6-11 Events

8. You can also further drill-down the check that the contents of these events look as you are expecting.
9. Select the event from CICS EP (double-click the time stamp).

10. The event payload (including the CICS context information) is in the any (xsd:any) slot. See Figure 6-12.

You can copy this into your favorite XML editor to further view and investigate the content. See Figure 6-13 on page 179.

Event Data	
List of all properties associated with the selected event.	
Name	Value
version	1.0.1
globalInstancelid	CE9E879B5BCDA68BBBA1DE7769D7441BD2
extensionName	
localInstancelid	
creationTime	2009-07-23T09:24:53.553+00:00
severity	
msg	
priority	
sequenceNumber	2
repeatCount	
elapsedTime	
reporterComponentId	
sourceComponentId / component	IBM CICS TS#4.1.0
sourceComponentId / subComponent	CICS EP
sourceComponentId / componentIdType	ProductName
sourceComponentId / instancelid	USIBMSC.EPRED4
sourceComponentId / application	
sourceComponentId / executionEnvironment	IBM z/OS
sourceComponentId / location	SC66
sourceComponentId / locationType	Hostname
sourceComponentId / processId	
sourceComponentId / threadId	
sourceComponentId / componentType	http://www.ibm.com/xmlns/prod/cics/eventprocessing
msgDataElement	
situation / categoryName	OtherSituation
situation / situationType / reasoningScope	EXTERNAL
situation / OtherSituation / any	<CICSApplicationEvent></CICSApplicationEvent>
any	<cics:event xmlns:cics="http://www.ibm.com/xmlns/prod/cics/evei

Figure 6-12 Event content

```

<cics:event xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
  <cics:context-info>
    <cics:eventname>Order</cics:eventname>
    <cics:usertag>V001</cics:usertag>
    <cics:networkapplid>USIBMSC.EPRED4</cics:networkapplid>
    <cics:timestamp>2009-07-16T09:24:53.553+00:00</cics:timestamp>
    <cics:bindingname>ShoppingMonitorBinding</cics:bindingname>
    <cics:capturespecname>LINKPOST</cics:capturespecname>
  </cics:context-info>
  <cics:payload-data>
    <data:payload
      xmlns:data="http://www.ibm.com/prod/cics/V001/Order">
        <data:CustomerNumber>+00005</data:CustomerNumber>
        <data:OrderNumber>+00028</data:OrderNumber>
        <data:StockId>+00006</data:StockId>
        <data:Quantity>+00010</data:Quantity>
      </data:payload>
    </cics:payload-data>
  </cics:event>

```

Figure 6-13 Event payload

Part 3



Scenarios



WebSphere Business Events scenario

In this chapter we run through the scenario described in Chapter 3, “Environment overview” on page 45.

7.1 Development setup, WebSphere Business Events tooling, and so forth

In this section we give an overview of our scenario, look at the tooling available, run through the scenario.

7.1.1 Scenario description: Interaction between multiple CICS events

In our scenario configuration, WebSphere Business Events and WMQ were both running on the same z/OS LPAR as CICS. Having WebSphere Business Events running on z/OS simplifies the WebSphere Business Events and WMQ configuration when compared with the same scenario using WebSphere Business Events on other platforms. That would require extra WMQ configuration. Having WebSphere Business Events running on z/OS allows WebSphere Business Events, which uses WAS as its runtime environment, to exploit the performance and scalability of the WAS on z/OS (zWAS) environment.

The CICS part of our scenario is described in Chapter 3, “Environment overview” on page 45.

In summary, the CICS COBOL application emits the events QueryStock, Order, Fulfill and Ship. These four events are defined in a single CICS event binding file and so are emitted through a single event adapter. For these events to be consumed by WebSphere Business Events, they are required to be sent from CICS to WebSphere Business Events through WMQ.

There is a one-to-one relationship between a CICS event binding file, the binding adapter, and the target WMQ queue. Also, remember that a CICS bundle can contain many event binding files. Therefore, to have each of the four events sent to separate WMQ queues, we would have defined each event in its own event binding file within a single event bundle.

The decision to have one WMQ queue for all events or to have a queue for each event type depends on the environment and throughput requirements. When a single WMQ queue is used, it reduces the JMS threads used in the WebSphere Business Events connectors and also reduces the memory usage of the connectors. The other advantage of using a single WMQ queue is that it reduces the WMQ administration required.

Multiple queues may be desirable when the events are coming into WebSphere Business Events at a high rate. So, if a single queue is used there is a possibility that messages will start to back up on the input queue,

Our scenario using WebSphere Business Events is based around the business wanting to know when customers query on a product several times but do not follow up by placing an order for the product, so that they can be proactive and offer a discount to encourage the customer to order the product. The business also wants to keep a record of orders that are placed by customers. The ability of WebSphere Business Events to look for patterns of events and for an absence of events over a time period, makes this possible.

Our WebSphere Business Events scenario defines a WebSphere Business Events interaction set that looks for a customer repeatedly querying on a product but not placing an order within a period of time. Therefore, we look for a customer querying on a single product more than three times in three minutes. When this occurs, we want to offer the customer a 10% discount. To do this, an action is sent from WebSphere Business Events to offer this discount. We could use the WebSphere Business Events eMail Connector to send an e-mail directly to the customer offering the discount, but for the sake of simplicity we send the offer out as an xml file using the WebSphere Business Events File System Connector.

We also define an interaction set to look for the customer placing an order and sent an action to the file system file out when the order was placed. An interaction set referencing the Order event was required to allow us to define the 'No Order' filter test (described next), to work. The two interactions sets created are shown in Figure 7-1.

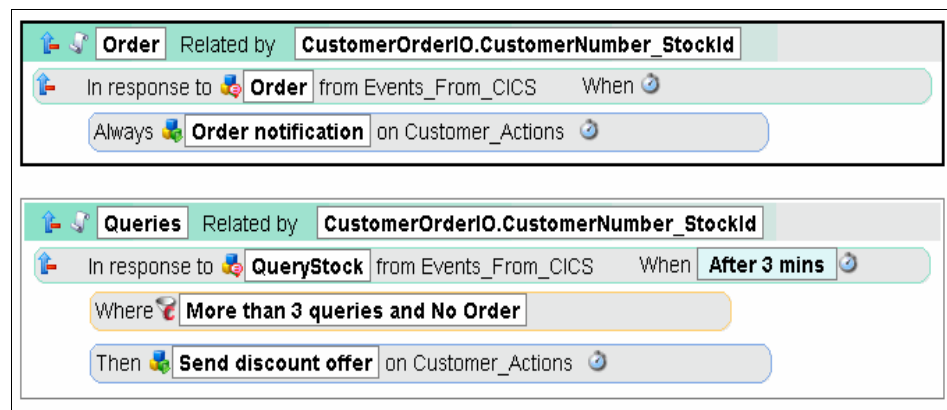


Figure 7-1 WBE interaction sets defined for our scenario

The first interaction set in Figure 7-1 shows that in response to every order that arrives on the `Events_From_CICS` touch point, we notify the customer of the order being placed by sending out an Order Notification action on the Customer Actions touch point.

The second interaction set looks for Query Events arriving on the Events_From_CICS touch point, but this time has the condition “More than three queries and No Order” defined. This condition, known in WebSphere Business Events as a filter, was implemented as two nested filters, as shown in Figure 7-2.

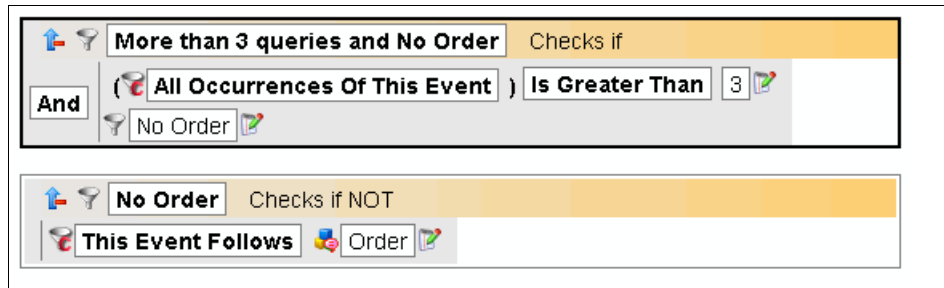


Figure 7-2 WBE Interaction Sets for the Shopping scenario

The second filter checks for the non-arrival of an Order event,

The first filter checks for more than three occurrences of this event in three minutes, and then calls the No Order filter.

The “Related by CustomerOrderIO.CUSTNO_STOCKID” entry in both interaction sets tells WebSphere Business Events to look for queries or orders where the CUSTNO_STOCKID Intermediate Object field in the CustomerOrderIO Intermediate Object matches over separate events. As a context relationship can only be based on a single field, the CUSTNO-STOCKID field is a concatenation of the incoming events, CUSTNO and STOCKID fields. These are concatenated using the JavaScript™ in Figure 7-3.

```
function pad(number, length) {
    var str = '' + parseInt(number, 10);
    while (str.length < length) {
        str = '0' + str;
    }
    return str;
}
var CustNum = pad(CustomerNumber, 5);
var StockIdent = pad(StockId, 5);
CustNum + StockIdent
```

Figure 7-3 CUSTNO_STOCKID concatenation

This Java script is placed in the CUSTNO_STOCKID Intermediate Object field definition in WebSphere Business Events Design Data, as shown in Figure 7-4.

The screenshot shows a configuration window for an intermediate object field. The field name is 'CustomerNumber_StockId' and its data type is 'String'. There is a checkbox for 'Record this data in History' which is currently unchecked. The 'Definition' section is expanded, showing the 'Type' as 'JavaScript'. The 'Source' is empty, with buttons for 'Insert source column ->' and 'Insert object field ->'. The 'Expression' field contains the following JavaScript code:

```
function pad(number, length) {  
    var str = "" + parseInt(number, 10);  
    while (str.length < length) {  
        str = '0' + str;  
    }  
    return str;  
}  
var CustNum = pad(CustomerNumber, 5);
```

Figure 7-4 JavaScript embedded into CUSTNO_STOCKID Intermediate Object definition

7.1.2 CICS parts: Event specs, export event schemas (WBE)

Once the CICS event specifications were defined in the Event Binding Editor within the CICS Explore, we exported them to xsd files so they could be imported in WebSphere Business Events Design Data, which is discussed later in this chapter.

To export the event specifications from the Event Binding Editor, open the event binding file and select the adapter tab, as shown in Figure 7-5.

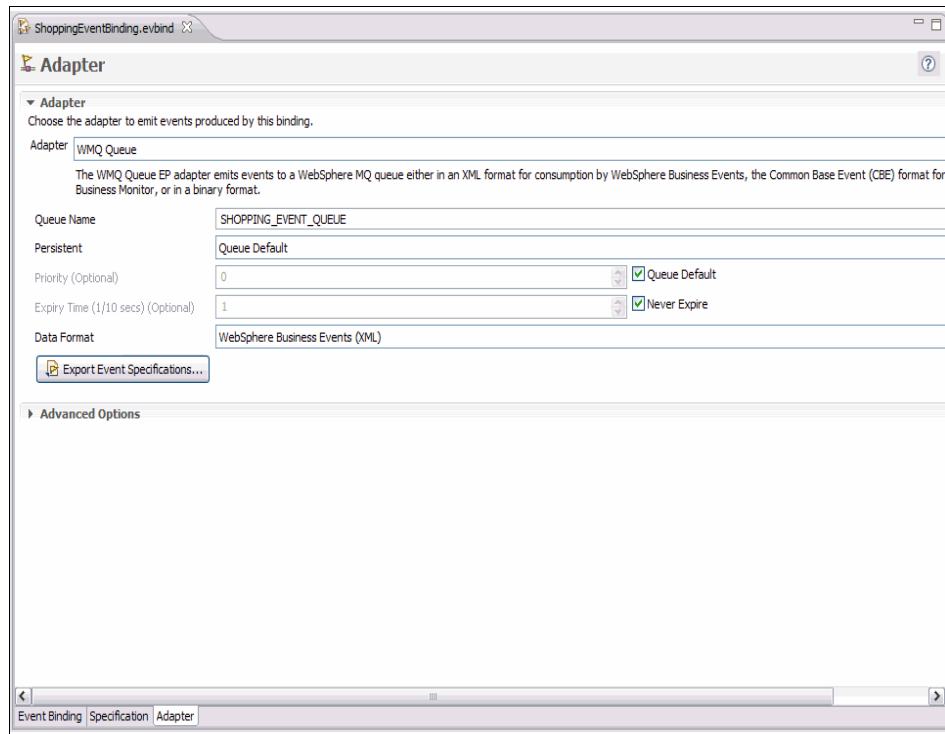


Figure 7-5 CICS Event Binding - AdapterC

Notice that for export to WebSphere Business Events, the data format was set to WebSphere BusinessEvents (XML). The other choices are CICS Flattened Event (Binary) and Common Base Event (XML). The CBE event format was used when we sent events directly to WBM. See Chapter 8, “WebSphere Business Monitor scenario” on page 213.

When we clicked on the **Export Event Specifications** button the Export Event Specifications dialog box (Figure 7-6 on page 189) was displayed.

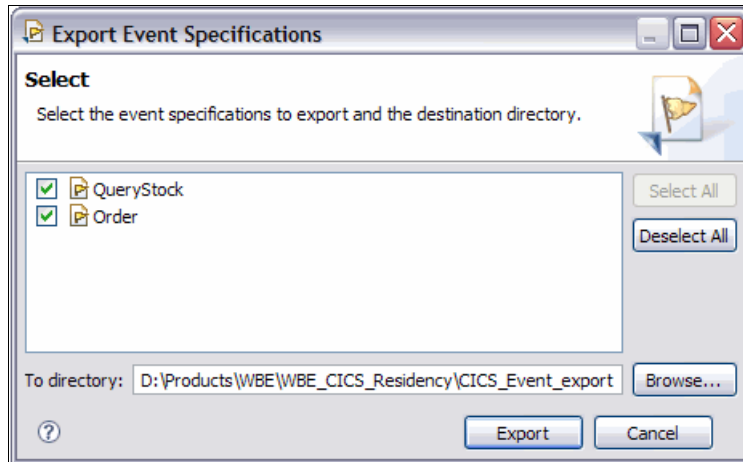


Figure 7-6 CICS Explorer, Event Binding Editor- Export Event Specification

We chose to export all four event specifications, and place them in the folder D:\Products\WBE\WBE_CICS_Residency\CICS_Event_export on the workstation.

In WebSphere Business Events Design Data, having defined a new touch point named Events_From_CICS, select **Import Event Like Schema From File** from the touch point context menu,. This allows us to select the event xsd files to import, as shown in Figure 7-7.

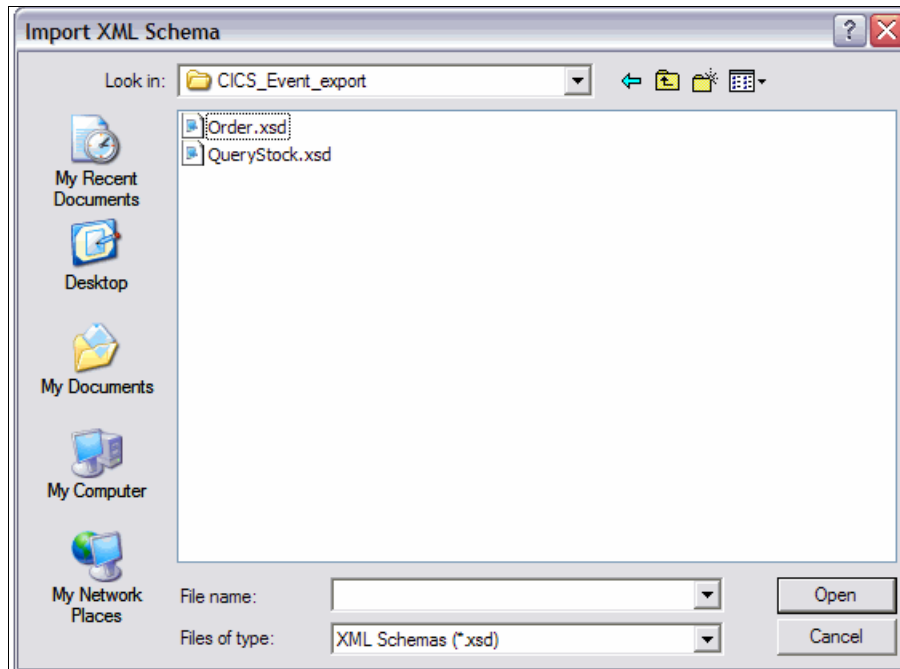


Figure 7-7 WBE Design Data Import XML Schema

We selected all four xsd files for import and clicked **Open**. This resulted in the message box shown in Figure 7-8 being displayed, confirming that four events had been imported.

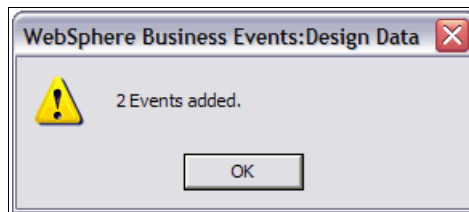


Figure 7-8 WBE Design Data xsd import confirmation

When the message box was closed (by clicking **OK**), the newly created events and events object tree, as shown in Figure 7-9, were created. In this panel, all the event specifications imported from the CICS Event Binding Editor are expanded and the Query_Data item has been selected and expanded to show the two data items that have been defined within the Query_Data object. The CUSTNO field has been selected and expanded, showing the xsd data type that has been imported from CICS Event Binding Editor. In this case, the CUSTNO event field was defined as numeric in the CICS Event Binding Editor.

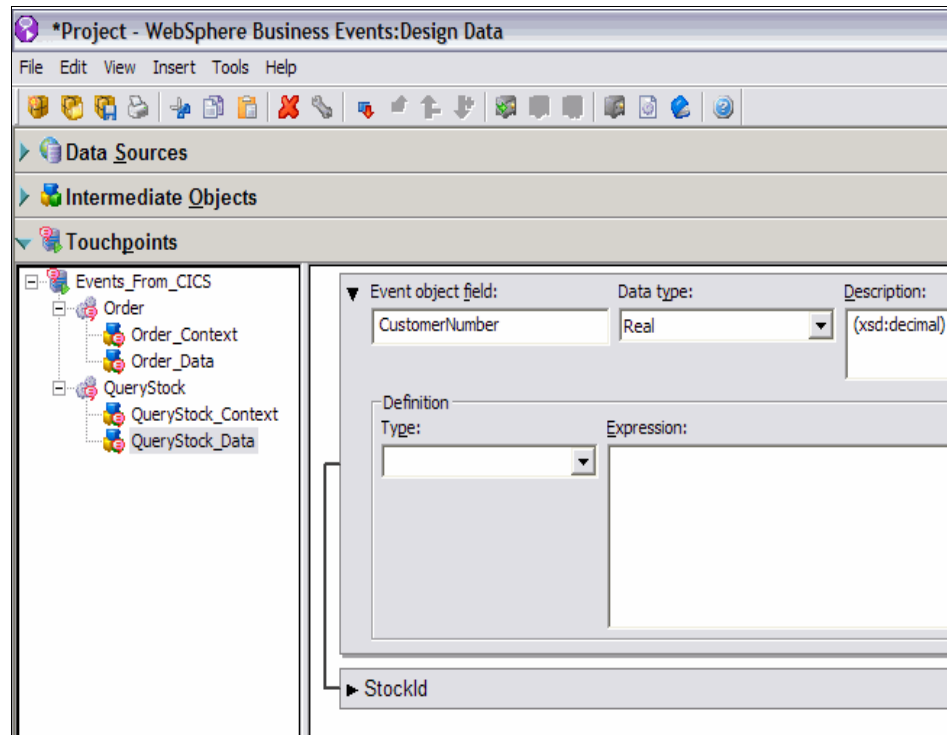


Figure 7-9 Result of importing CICS Event Specifications into WBE Design Data

7.1.3 WebSphere Business Events development tooling

WebSphere Business Events tooling was designed with two classes of users in mind, the IT developer and the business user.

IT developers use the Design Data tool to create events, actions, intermediate objects, and data source definitions. The skills required to achieve this are primarily technical in nature and require understanding of the protocols and formats of data. Once created, these objects are loaded into a common repository. The WebSphere Business Events Repository is part of the runtime

database used by WebSphere Business Events. In our configuration we used DB2 running on the same z/OS LPAR for our relational database. When using z/OS for the WebSphere Business Events runtime, DB2 on z/OS is the only option for the runtime database.

When using WebSphere Business Events on other platforms, other database engines can be used (such as Oracle or Microsoft SQL server). The details of which database can be used on which environment are listed on the WebSphere Business Events system requirements page, available at the following Web page:

<http://www-01.ibm.com/software/integration/wbe/requirements/>

Business users, through the design tool, retrieve objects from the repository to define the business conditions that link the events and actions (such as: if event A and event B happen with three days and this filter evaluates to true, then start this specific action).

The two WebSphere Business Events development tools, their usage, and how they relate to the WebSphere Business Events runtime is shown in Figure 7-10.

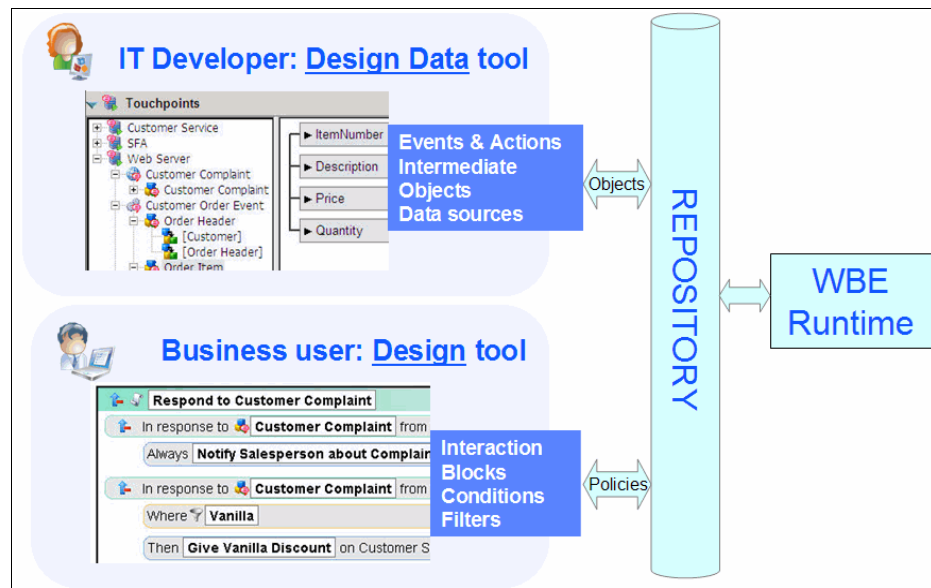


Figure 7-10 WBE Development Tooling

Both Design Data and Design use XML files to store projects. Once the project artifacts are ready to be shared amongst other users and made available to the WebSphere Business Events runtime, they are checked into the WebSphere Business Events repository.

7.1.4 WMQ setup and WebSphere Business Events configuration for WMQ

CICS events for consumption by WebSphere Business Events are sent from CICS to WebSphere Business Events through a WMQ queue. WebSphere Business Events uses its message queue connection to read the message from WMQ and pass it on to the WebSphere Business Events runtime for processing.

We define a WMQ local queue on Queue Manager MQ8G with all the default properties as in Figure 7-11.

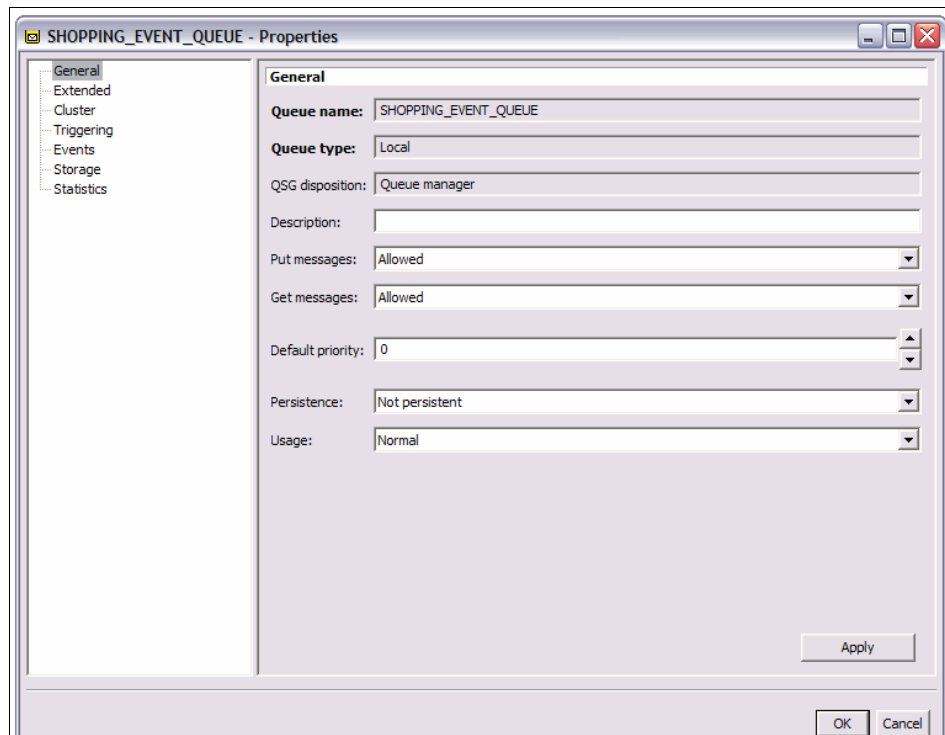


Figure 7-11 WMQ local queue definition

As can be seen in Figure 7-11, we used the WMQ Explorer tool to define the queue on the z/OS-based queue manager, but could have used the WMQ Operations and Control panels in ISPF.

The Event Binding Editor within CICS Explorer was used to define the Query, Order, Fulfill, and Ship events in a event binding file, as shown in Figure 7-12.

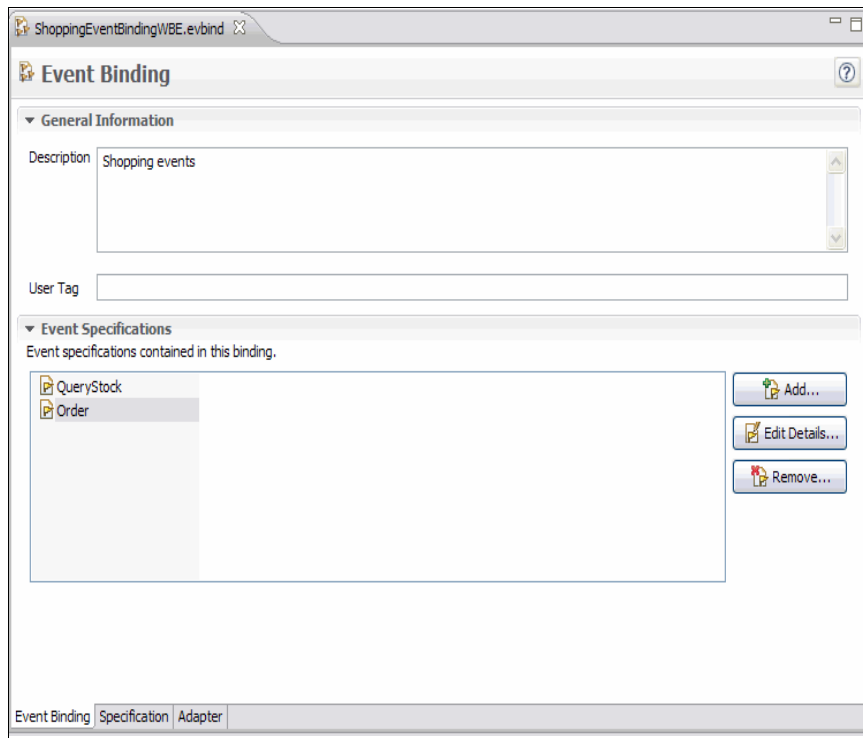


Figure 7-12 CICS Explorer, Event Binding

Looking at the Adapter tab (Figure 7-13 on page 195) in the event binding shows that we defined the adapter as a WMQ adapter (the other choices are TS Queue, Transaction Start or Custom (User Written)), and the queue name as SHOPPING_EVENT_QUEUE. Note that the WMQ queue manager name will be defined by the connection between CICS and WMQ. In our case, our CICS region connected to the queue manager MQ8G at CICS start up.

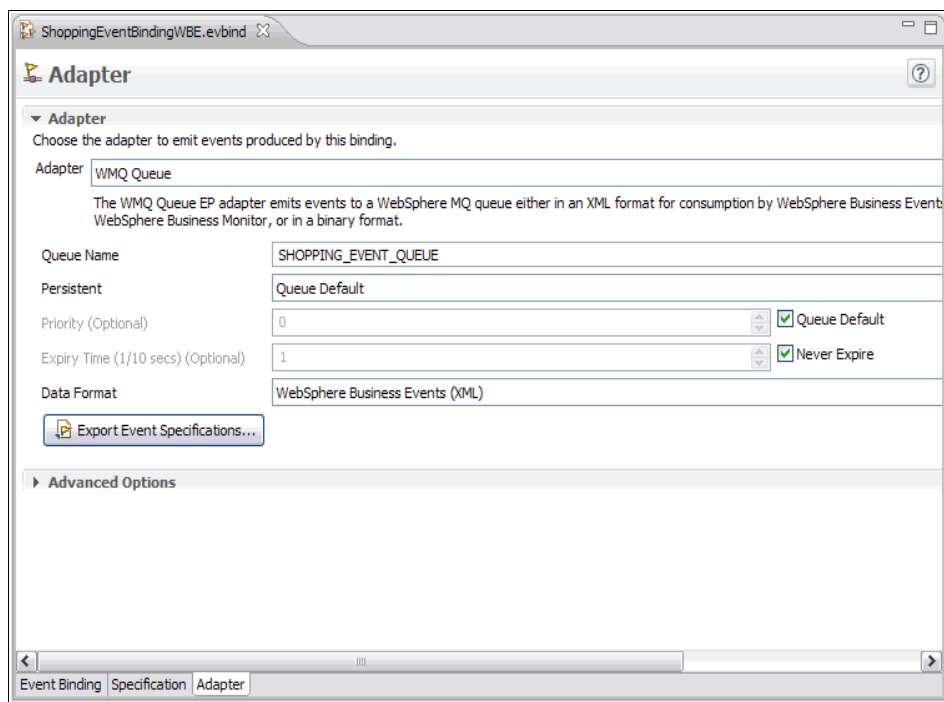


Figure 7-13 CICS Event Binding - Adapter

Once the WMQ queue was defined and before we did any configuration in WBE, we defined the JMS Queue Connection Factory and Queue in JNDI. We used the file system JNDI context and located it in the directory `/u/wqadmin/JNDI`. Two options exist for managing the JNDI repository:

- ▶ Using the command line JMSAdmin tool supplied with WMQ
- ▶ Using the JMS Administration facilities provided in WMQ explorer.

We chose the latter method and then ftp'd the resulting.bindings file to the mainframe using binary mode and placed in the directory `/u/wqadmin/JNDI`. We had to use ftp to transfer the.bindings file as a binary file as WMQ Explorer cannot connect directly to the z/OS UNIX system services file system. The .bindings file can be found under the root of the JNDI filesystem namespace. In the WMQ Explorer, we created a Queue Connection Factory (QCF) named WebSphere Business EventsQCF. The settings were as shown in Figure 7-14 on page 196.

```

InitCtx> dis qcf(*)
QCF(WBEQCF)
  MSGRETENTION(YES)
  QMANAGER(MQ8G)
  TARGCLIENTMATCHING(YES)
  TEMPMODEL(SYSTEM.DEFAULT.MODEL.QUEUE)
  USECONNPOOLING(YES)
  POLLINGINT(5000)
  RESCANINT(5000)
  TRANSPORT(BIND)
  TEMPQPREFIX()
  SYNCPOINTALLGETS(NO)
  MAPNAMESTYLE(STANDARD)
  CONNOPT(STANDARD)
  VERSION(6)
  MSGBATCHSZ(10)
  FAILIFQUIESCE(YES)

```

Figure 7-14 JNDI Queue Connection Factory for WBEQCF

We also created a JMS Queue with the attributes in Figure 7-15.

```

InitCtx> dis q(*)
Q(WBEQ)
  FAILIFQUIESCE(YES)
  QUEUE(SHOPPING_EVENT_QUEUE)
  QMANAGER(MQ8G)
  PERSISTENCE(APP)
  CCSID(1208)
  TARGCLIENT(JMS)
  ENCODING(NATIVE)
  PRIORITY(APP)
  EXPIRY(APP)
  VERSION(6)

```

Figure 7-15 JNDI JMS Queue definition

In WebSphere Business Events Design Data, we defined the connection to the WMQ queue through the event properties for the CICS query event. The event properties are assessed by selecting **Event Properties** from the event context menu, as shown in Figure 7-16 on page 197.

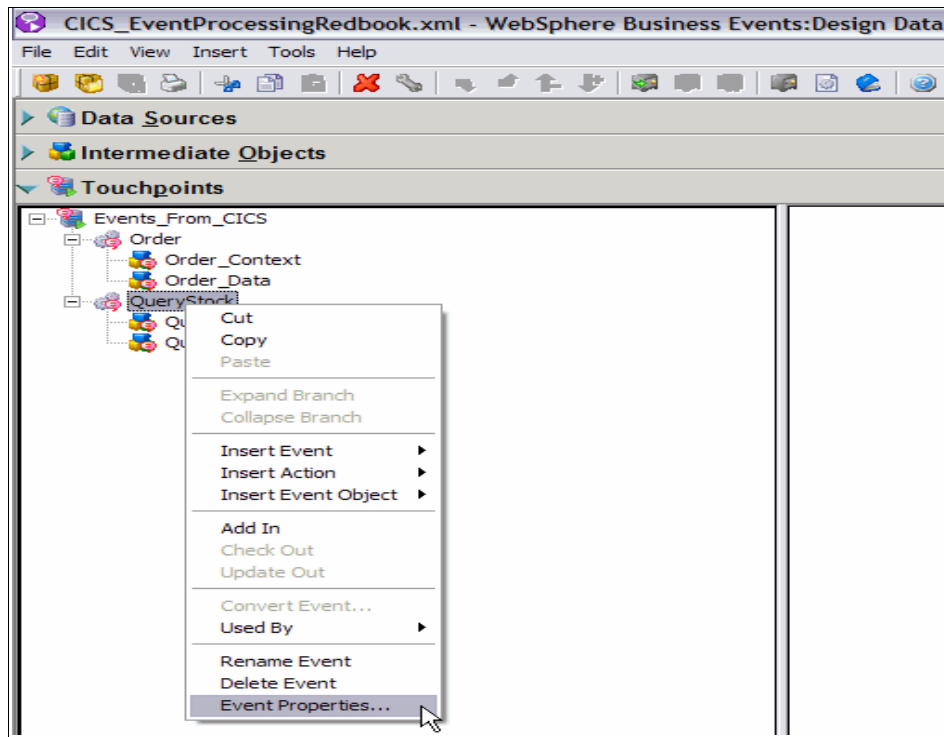


Figure 7-16 Accessing the WBE Event properties in Design Data

On the resulting “Event Query Properties” dialog box, we selected the **Message Queue Connection** radio button and clicked **Configure**. This gave us the “Message Queue Event Connection” dialog box, which allowed us to provide the JMS Queue name, WBEQ.

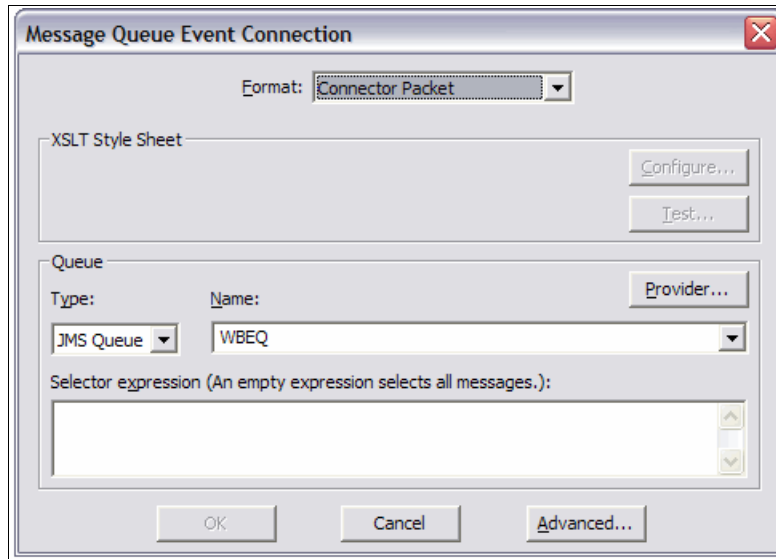


Figure 7-17 Message Queue Event properties

The **Provider** button is used to define the JNDI provider for the JMS resources. Clicking this presented the “Provider” dialog box, shown in Figure 7-18. We used the file system provider that we set up previously, and the WBEQCF Queue Connection Factory created in that JNDI.

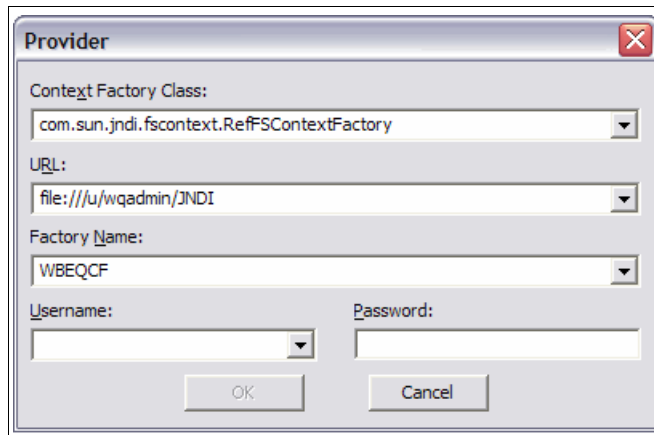


Figure 7-18 JNDI Provider connection dialog

Notice that the user ID and password were not defined, as JNDI security was not enabled at this point.

7.1.5 WebSphere Business Events Configuration for actions

To get the discount offer to the customer, we had to configure the action connector on the “Send discount offer” message. The action connector is accessed the same way as the event connector discussed in 7.1.4, “WMQ setup and WebSphere Business Events configuration for WMQ” on page 193, except this time we selected **File System Connection** radio button, as shown in Figure 7-19.

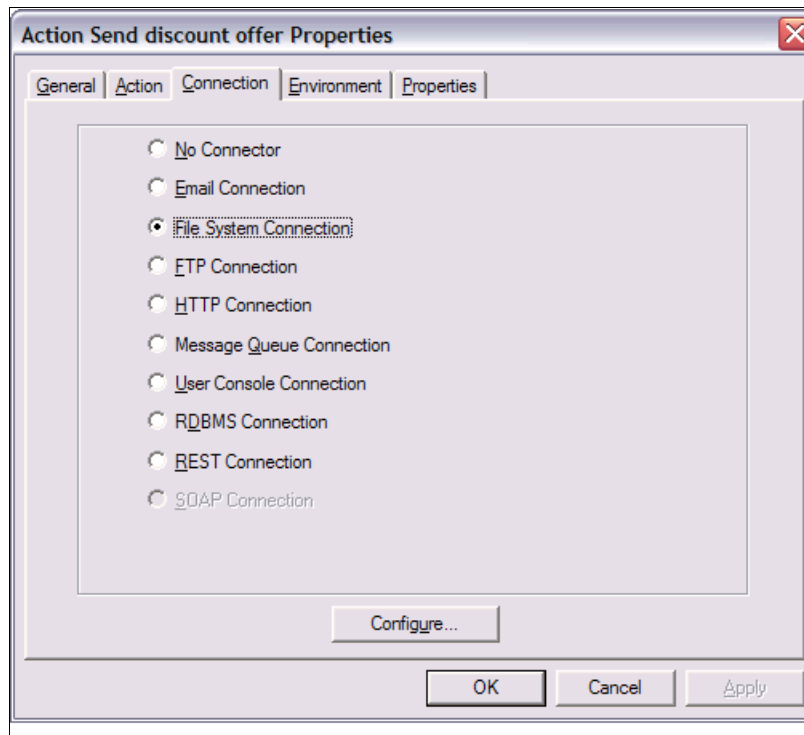


Figure 7-19 Accessing the action's connection configuration

Clicking **Configure** takes us to the “File Action Connection” dialog box shown in Figure 7-20.

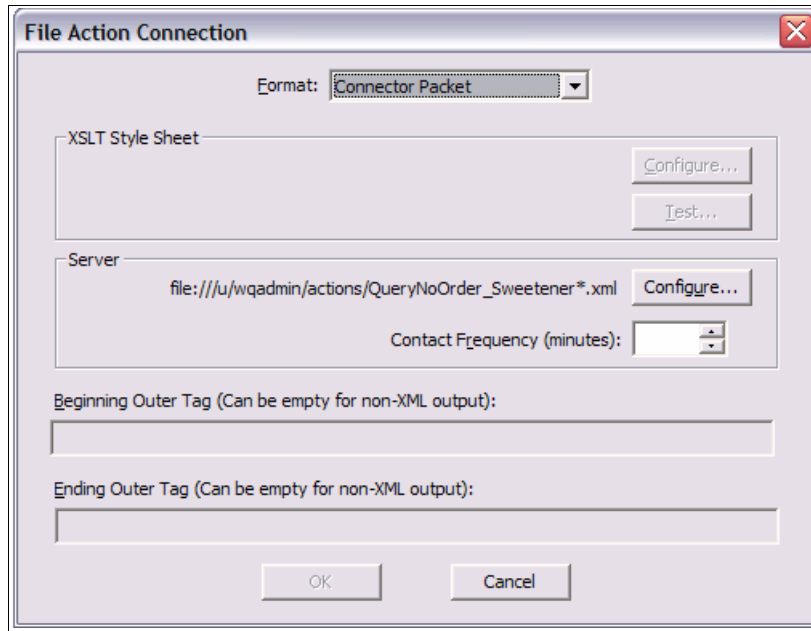
The image shows a Windows-style dialog box titled "File Action Connection" with a red close button in the top right corner. The dialog is divided into several sections. At the top, there is a "Format:" label followed by a dropdown menu currently set to "Connector Packet". Below this is a section for the "XSLT Style Sheet" with a large empty text area and two buttons, "Configure..." and "Test...", to its right. The next section is labeled "Server" and contains a text field with the value "file:///u/wqadmin/actions/QueryNoOrder_Sweetener*.xml" and a "Configure..." button to its right. Below the server path is a "Contact Frequency (minutes):" label followed by a small spinner control. The bottom section contains two labels: "Beginning Outer Tag (Can be empty for non-XML output):" and "Ending Outer Tag (Can be empty for non-XML output):", each followed by a large empty text area. At the very bottom of the dialog are "OK" and "Cancel" buttons.

Figure 7-20 File System Connection configuration dialog box

The only thing we need to check in this dialog box is that the “Format” drop down menu is set to “Connection”. Notice that the Server box names the location and the name of the file to be sent out as an action. These values are configured by clicking **Configure**, which takes us to the “Server” dialog box shown in Figure 7-21 on page 201.

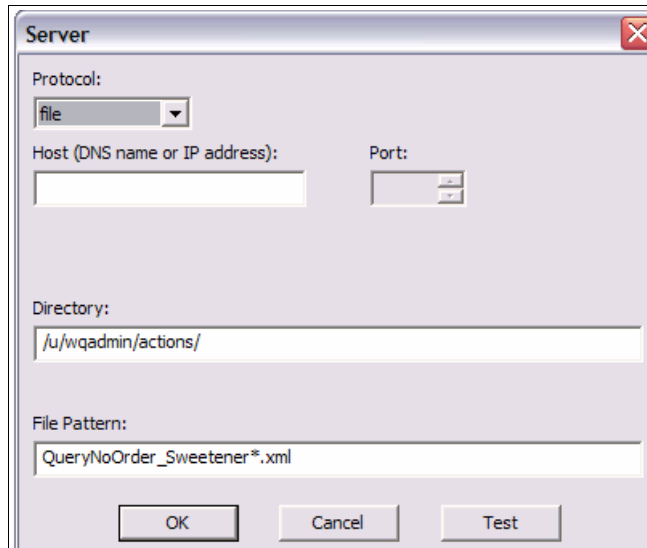


Figure 7-21 Server configuration dialog for File System Connection

The directory that the file will be sent to is set to `/u/wqadmin/actions/` and the file pattern is set to `QueryNoOrder_Sweetener*.xml`. This file pattern means a file will be generated with the prefix `QueryNoOrder_Sweetener`. The asterisk means that WebSphere Business Events will generate the following part of the name.

7.1.6 Running the WebSphere Business Events scenario

Before attempting to run the scenario end-to-end from CICS, we made sure that the WAS server that hosted WebSphere Business Events running on our z/OS system was running. To start our WAS system, we used the following start command:

```
S WQ6661,JOBNAME=WQ6661,ENV=CL6661.ND6661.WQ6661
```

The message BB000019I INITIALIZATION COMPLETE FOR WEBSPHERE FOR Z/OS CONTROL 517 in the log indicated that the WAS server started successfully. In the DA view of SDSF we saw four started tasks running for the WAS server, as shown in Figure 7-22.

Display Filter View Print Options Help											

SDSF	DA	SC66	(ALL)	PAG	0	CPU/L/Z	5/	2/	0	LINE 1-4 (4)	
COMMAND INPUT ==>>										SCROLL ==>>	CS
PREFIX=WQ6* DEST=(ALL) OWNER=** SORT=JOBNAME/A SYSNAME=*											
NP	JOBNAME	StepName	ProcStep	JobID	Owner	C	Pos	DP	Real	Paging	SI
	WQ6661	WQ6661	BB0CTL	STC11287	WQACRU	NS	F2	56T	0.00	0.0	
	WQ6661A	WQ6661A	BB0SR	STC11291	WQASRU	IN	F2	89T	0.00	0.0	
	WQ6661D	WQ6661D	BB0DAEMN	STC11289	WQACRU	NS	FE	6813	0.00	0.0	
	WQ6661S	WQ6661S	BB0SR	STC11290	WQASRU	IN	F2	160T	0.00	0.0	

Figure 7-22 Started tasks for WAS

WQ6661 is the Control Region, WQ6661S is the Servant region (that runs the WebSphere Business Events and other WAS applications), WQ6661A is the server Adjunct region (which supports the WAS (SI Bus) messaging), and WQ661D is the z/OS Location Service Daemon.

Logging on to the Integrated Solutions Console (ISC) (Figure 7-23 on page 203), we see that the wberuntimeear application is installed and started by selecting Enterprise Applications from the tasks list on the left hand side of the panel.

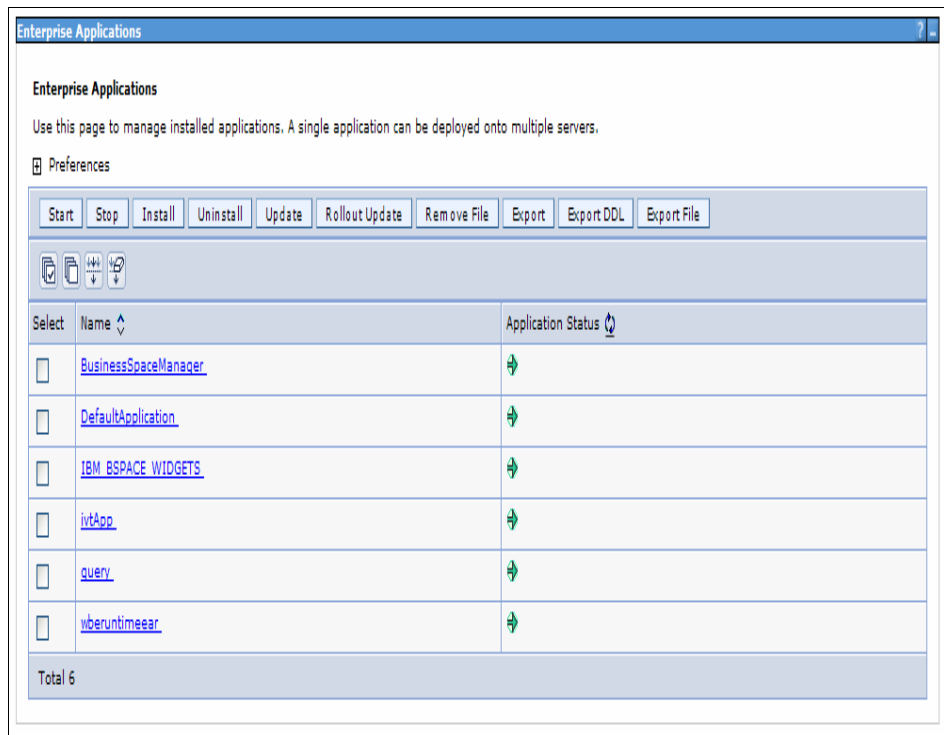


Figure 7-23 Wberuntimeear installed and started in WAS.

Once we had confirmed that WebSphere Business Events was up and running, we deployed the WebSphere Business Events project to the runtime through the repositories facility in WebSphere Business Events Design Data. To do this we started Design Data, opened our project file and then selected **Tools Repositories** from the menu bar. Next, we chose the WebSphere Business Events runtime tab in the Repository dialog box and clicked the **Login** button. Finally, in the Repository Properties dialog box, we supplied the host and port, as shown in Figure 7-24.

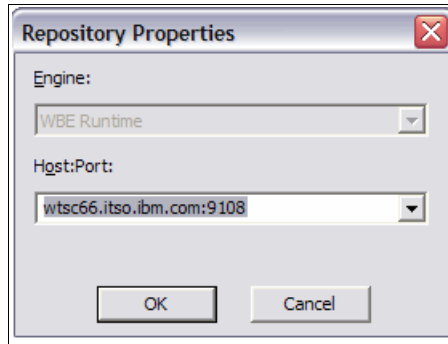


Figure 7-24 WBE Design Data - Repository Properties

In our case the host name was set to `wtsc66.itso.ibm.com` and the port set to 9108 (the HTTP port for the WAS server).

Next, we were prompted for the WebSphere Business Events administrator user ID and password and clicked **OK**. When we logged on for the first time after WebSphere Business Events had been installed, selecting “director on `wtsc66.itso.ibm.com`” tab showed that the repository was empty. To add the contents of our project to the runtime repository we clicked the Project tab, selected all the assets in the project and then clicked **Add-IN**. Once all the Add-In have been selected, and after a short pause, all the assets were added to the “director on `wtsc66.itso.ibm.com`” tab.

Next, we closed the Repository dialog box and exited Design Data, saving the project file when prompted to do so.

Having added the assets to the WebSphere Business Events runtime repository, the next thing we had to do was refresh the runtime with the new assets. This is the equivalent of a new copy. To do this, we had to log on to the WebSphere Business Events Administrators Console. On our system the WebSphere Business Events Admin Console was accessed through the URL:

`http://wtsc66.itso.ibm.com:9108/wbe/common/login.jsp`

See Figure 7-25. Notice that “Tool drop down menu is set to “Administration”. The other options are “User Console” or “Business space”.

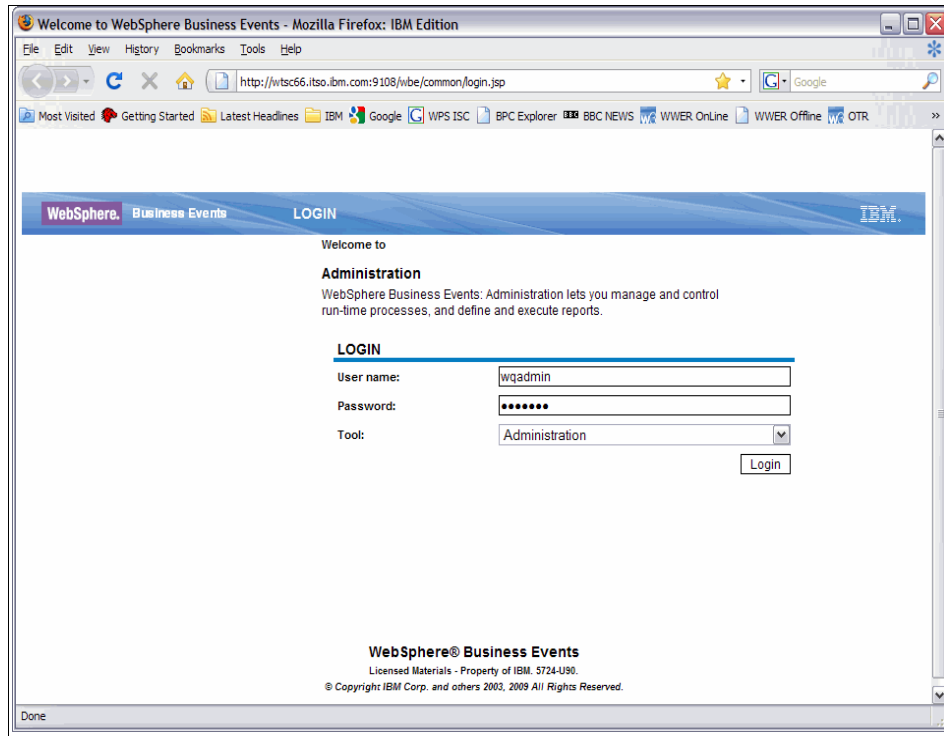


Figure 7-25 WBE Administrator Console login

Once we had successfully logged on, we clicked **Console** at the top left hand corner of the panel. On the console page we choose **Currently loaded interaction blocks** from the ‘Show’ drop-down menu, which showed that nothing was loaded into the runtime. See Figure 7-26.

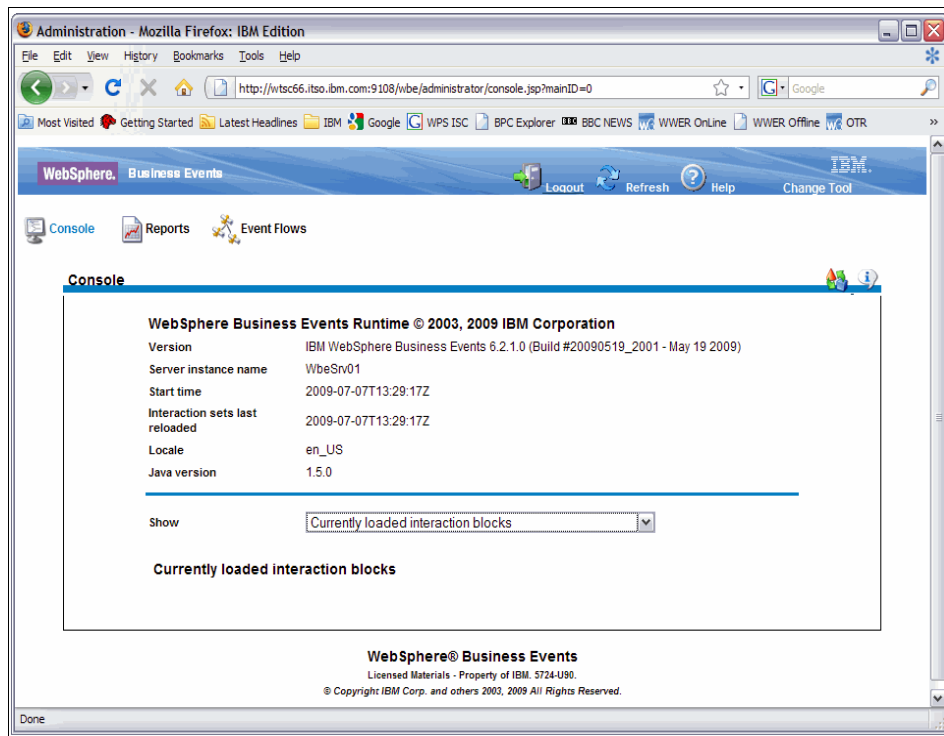


Figure 7-26 WBE Admin Console: Before loading new assets

To load the new assets from the repository, we clicked the reload project icon (the left of the two icons above the blue line on the right side). (If you hover the mouse pointer over it the pop-up message will say “reload project”.) While the reload was in progress, the message “reloading repository” was displayed beneath the reload icon. Once the reload is complete we receive message “BEER3043I” this confirms the repository was successfully reloaded.

We then checked to see which touchstones, events, and actions had been loaded by choosing this option from the “Show” drop down menu. We checked which interaction sets had been loaded by selecting “Currently Loaded Interaction sets”. The result was as shown in Figure 7-27 on page 207.

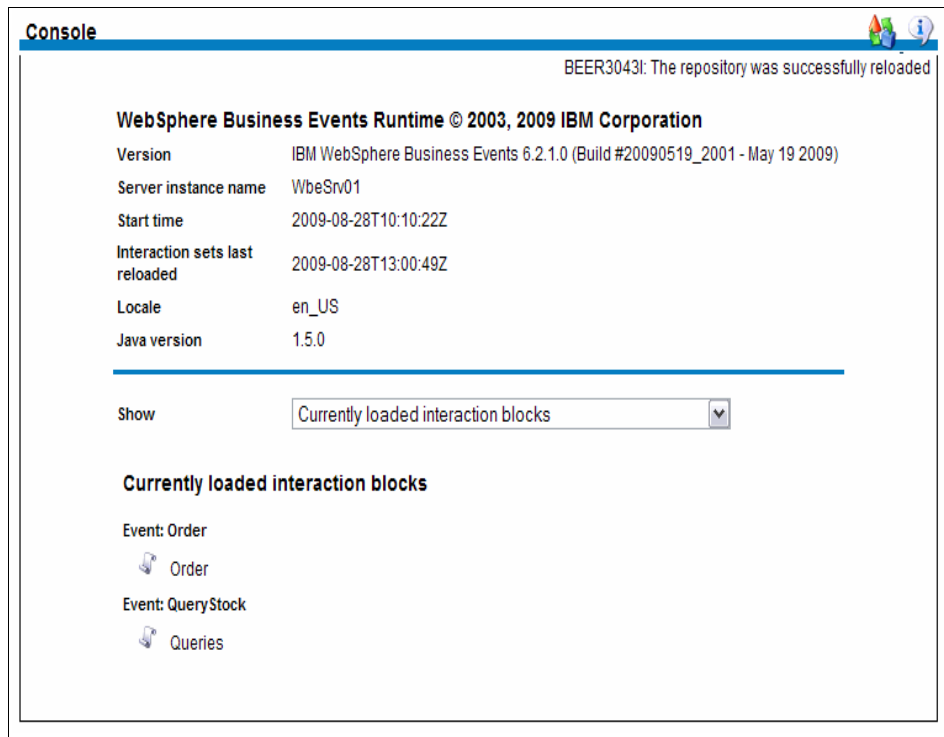


Figure 7-27 Scenario Interaction sets loaded into WBE

At this point we logged out of the WebSphere Business Events Admin Console.

As our CICS Events were being sent to WebSphere Business Events through WMQ, we made sure that the WMQ Queue Manager was started and the Queue SHOPPING_EVENT_QUEUE defined.

The next step was to start the WebSphere Business Events connectors. We submitted the JCL shown in Figure 7-28. The connectors implement the connections defined on the event and action properties in WebSphere Business Events Design Data.

```
//WQ6661C JOB ,,NOTIFY=&SYSUID,MSGCLASS=H
/*JOBPARM S=SC66
/*****
/* JCL TO START THE WBE CONNECTORS
/*****
/*
//CONNECT EXEC PGM=BPXBATCH,REGION=0M,TIME=NOLIMIT,
// PARM='sh /usr/lpp/wbeBE1/V6R2M1/director/bin/connectors_z0S.sh -nc'
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDENV DD PATH='/u/wqadmin/wbe/env.profile'
//SYSMDUMP DD SYSOUT=*
/*
```

Figure 7-28 JCL used to start WBE connectors

The env.profile file referenced by the STDENV DD statement contains the environment variables for the connectors. In our case, the env.profile contained the values shown in Figure 7-29.

```
WBE_CONNECTORS_CONSOLE_ENCODING=IBM-1047
WBE_CONNECTORS_FFDC_DIR=/u/wqadmin/wbe/ffdc
WBE_INSTALL=/usr/lpp/wbeBE1/V6R2M1
WBE_HOME=/u/wqadmin/wbe
WBE_WAS_HOME=/WebSphereBE1/V6R1/AppServer
CLASSPATH=/usr/lpp/mqm/mq8g/java/lib/com.ibm.mq.jar:/usr/lpp/mqm/mq8g/java/lib/com.ibm.mqjms.jar:/usr/lpp/mqm/mq8g/java/lib/jms.jar:/usr/lpp/mqm/mq8g/java/lib/dnbc core.jar
WBE_CONNECTORS_JVM_ARGS=-Djava.library.path=/usr/lpp/mqm/mq8g/java/lib
STEPLIB=MQ600.SCSQAUTH:MQ600.SCSQANLE
```

Figure 7-29 env.profile contents for WBE connectors

As the connectors started, messages like the one in Figure 7-30 appeared on the jobs STDOUT.

```
BEER0561I: The technology connectors are waiting for wberuntimeear.  
BEER0644I: The technology connectors established communication with  
wberuntimeea  
BEER0590I: Connector repository checkpoint: 2009-07-07 14:43:07.437  
BEER0508E: The file was not found java.io.FileNotFoundException:  
/u/wqadmin/wbe/  
BEER1853I: FFDC output directory cannot be established. Information will be  
writ  
BEER0642I: The file system module is monitoring action: Queries_NoOrder  
BEER0642I: The file system module is monitoring action: Order  
BEER0642I: The file system module is monitoring action: Query  
BEER0616I: The JMS module is monitoring event: Query  
BEER0611I: Initialization is complete  
BEER0670I: JNDI provider URL: corbaloc:iiop:wtsc66.itso.ibm.com:9101
```

Figure 7-30 WBE Connectors startup messages

Notice the message stating that the JMS module is monitoring event Query. This shows that the JMS connector is looking for a WMQ message arriving on our WMQ input queue from CICS. When we looked at this queue (SHOPPING_EVENT_QUEUE) in WMQ explorer we saw that the open input count was 1, and the status of the queue showed the app name as WQ6661C5, one of the connector job processes.

Our WebSphere Business Events scenario was looking for a single customer to query on a single product more than three times in three minutes. To check for events arriving into WebSphere Business Events, we logged on to the WebSphere Business Events administrator console and clicked on the “Reports” icon on the top left corner of the panel.

To simulate this we logged onto our CICS region and started the MENU transaction. We then entered a customer number (for example, 00001) and pressed PF1 to enter the query list menu. From here we selected an item (00010) to query on by entering any character in the “S” column. The “Query Item” panel displayed, showing the items details, including Stock Level. Pressing enter here returned us to the query list, where we repeated the same process for the same item three times.

We logged in to the WebSphere Business Events Administrator Console to check that QueryStock events that had been sent from CICS had arrived and had been detected by WebSphere Business Events. To do this we selected the Reports option and chose “Events by Touch point by time”, left the “Available Events” to default, and clicked **Generate Report**, which gave the result shown in Figure 7-31.

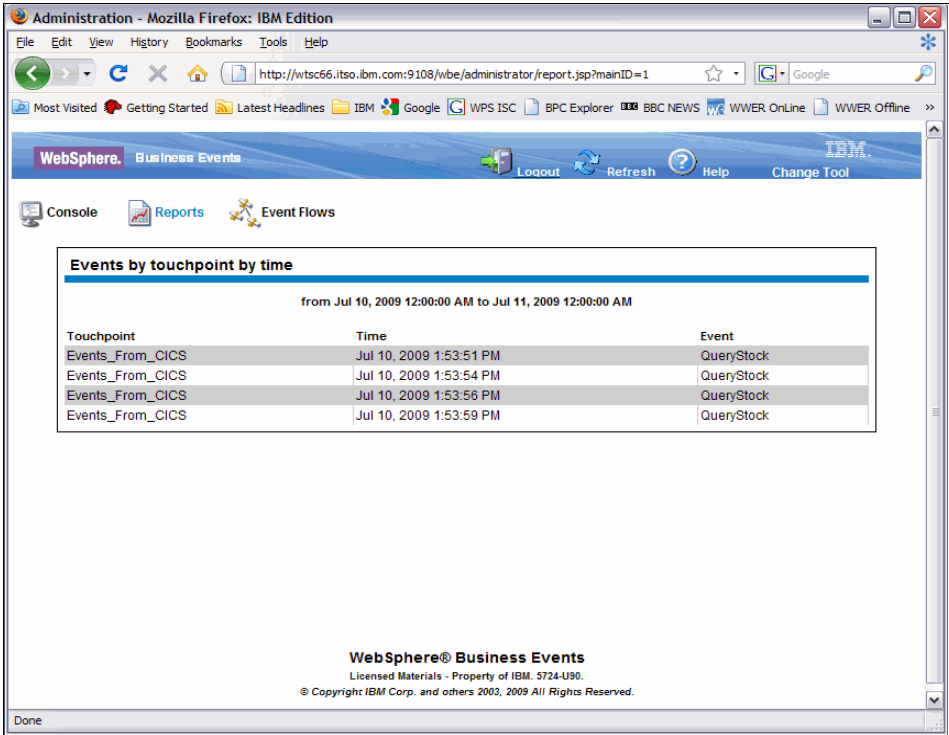


Figure 7-31 WBE report showing the QueryStock Events have arrived into WBE

After four minutes we ran the same report again and got the result shown in Figure 7-32 on page 211.

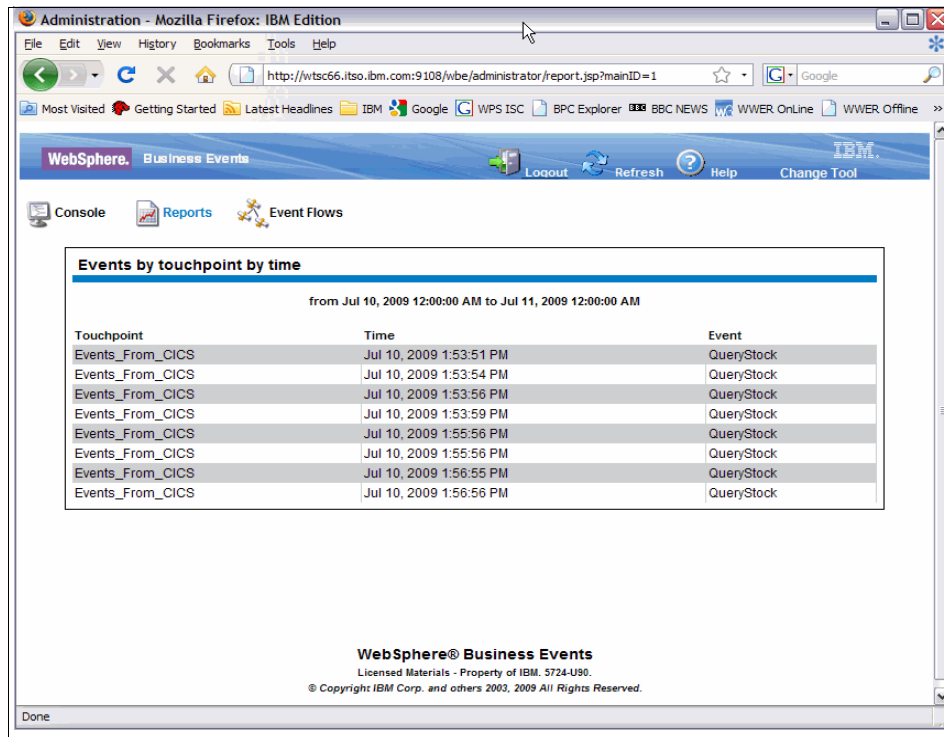


Figure 7-32 WBE report showing events after 3 mins

We can see from Figure 7-32 that WebSphere Business Events shows eight events arriving from CICS. Notice that the second set of four events arrived three minutes after the original four. Why are these extra events here? The second four QueryStock events are produced internally by WebSphere Business Events when the time passed event for the first four is evaluated after three minutes. This does not mean that our set of four QueryStock events occurs twice. The second set is only used internally by WebSphere Business Events.

See the Actions report shown in Figure 7-32 on page 211.

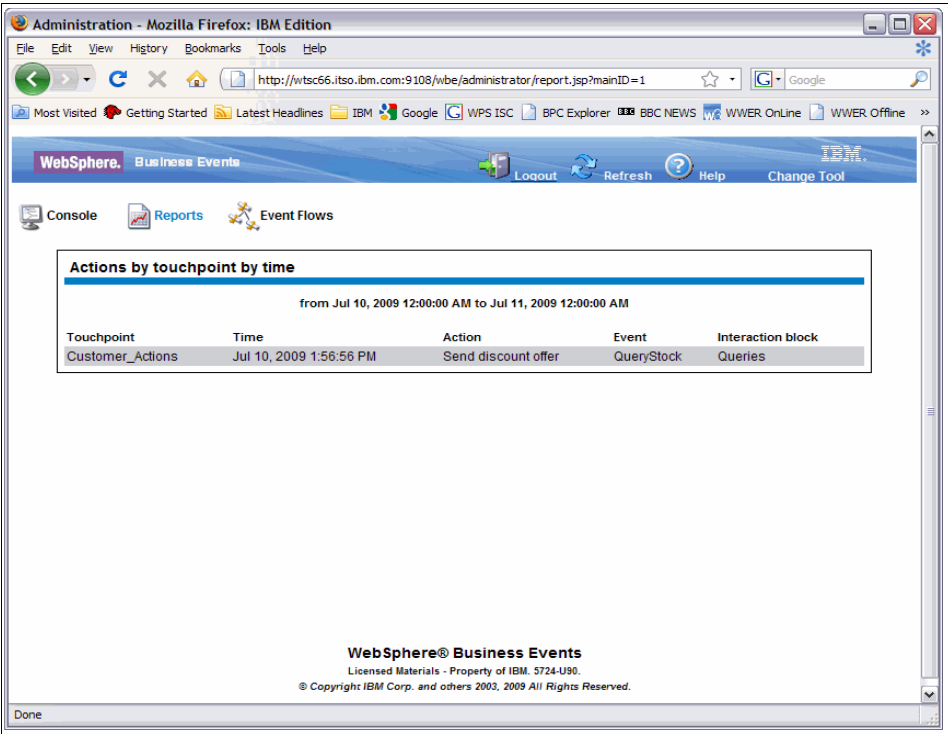


Figure 7-33 WBE Action report showing Send Discount offer action

This shows that the discount offer has been sent out to the customer. We then look in the `/u/wqadmin/actions/` directory in UNIX system services on z/OS. We see that a file has been created to send the discount. See Figure 7-34.

```
CICSR6:/u/wqadmin/actions: >ls -l
total 2
-rw-rw----  1 CICSR6  WQCFG          720 Jul 10 08:56
QueryNoOrder_Sweetener969825BD11E06D5111DE.xml
CICSR6:/u/wqadmin/actions: >
```

Figure 7-34 File sent out as a result of WBE action



WebSphere Business Monitor scenario

In this chapter we take the events that have been generated by CICS and process them in the WebSphere Business Monitor (also known in this chapter as the Monitor). Once these events have been consumed by the Monitor we will be able to see the analysis of our shopping process in the Business Space dashboards.

8.1 Scenario description

Our scenario for this business activity monitor centers around the shopping application, which is described in Chapter 3, “Environment overview” on page 45.

In our scenario the main focus of our business activity monitoring (BAM) is the customer. We will be monitoring the stock querying and buying behavior of our customers. We will also be tracking key performance indicators (KPIs), which are the metrics that we use to quantify and measure business performance against strategic and operational business targets (such as tracking actual order fulfillment durations against targets).

8.2 CICS parts: Event specs, export event schemas

As in Chapter 5, “Generating events” on page 91, the ShoppingMonitorBundle was created for the events that are to be sent to the WebSphere Business Monitor, with an event binding of ShoppingMonitorBinding. The event specifications for QueryStock, Order, Ship, and SendOrder were exported. These are the event schemas that we use to define the events to be processed by the monitor model that we build in 8.4, “Create monitor model” on page 235.

8.3 WMQ set up, WMQ on z/OS through SIB to CEI

In this section, we demonstrate the steps necessary to enable the WebSphere Business Monitor server to receive Common Base Events (CBEs) using WebSphere MQ as the transport mechanism. We begin by installing an EAR file. Then we carry out the necessary CEI configuration steps. We complete the interconnection with configuration steps on the WebSphere MQ on z/OS side. We then show how to verify the configuration was successful. These are one-time set-up steps and only need to be completed the first time the WebSphere Business Monitor server is being configured to consume CICS events through WebSphere MQ.

8.3.1 Installing the mediation EAR file

Note: Details on how to install WebSphere Business Monitor can be found in the product information center, at the following Web page:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.btools.help.monitor.install.doc/install/intro.html>

First, install the mediation EAR file provided with WebSphere Business Monitor. Install on the same server (or cluster) where CEI is configured in the local monitor cell. If you are not sure which server this is, first check the details of the CEI server. Perform the following steps to install the mediation EAR file:

1. Using the Administrative Console, expand **Service integration** and click **Buses** (Figure 8-1).



Figure 8-1 Service integration menu in the WebSphere Business Monitor administrative console

2. Click **CommonEventInfrastructure_Bus**.
3. Click **Bus Members**.

There should be a single bus member. Its name will include the node and the name of the server where the mediation EAR needs to be installed. This is shown in Figure 8-2. In our example, the server name is server1.

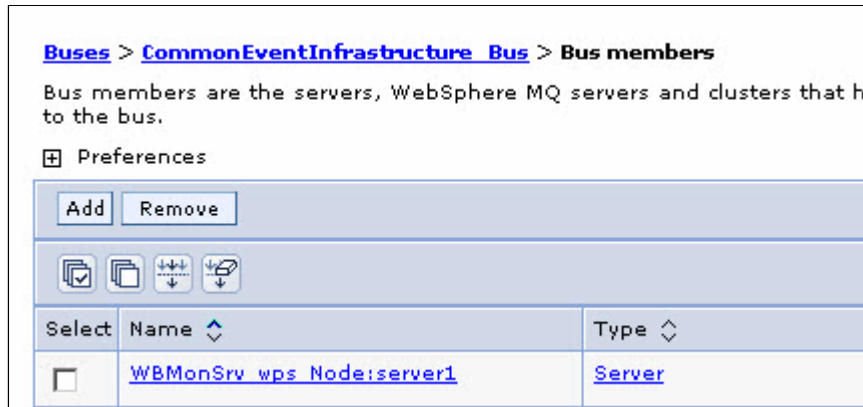


Figure 8-2 Server where the CEI is installed

Next, perform the following steps to install the mediation application:

1. Using the WebSphere Business Monitor server Administrative Console, expand **Applications** and click **Install New Application** (Figure 8-3).

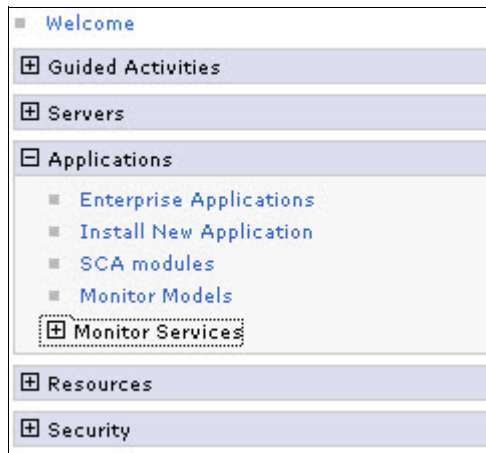


Figure 8-3 Install New Application menu

2. In the “Preparing for the application installation” panel (Figure 8-4), perform the following steps:
 - a. Ensure that the “Local file system” radio button is selected.
 - b. Click **Browse** and navigate to
<was_root>/scripts.wbm/CEIMQ/MQtoCEIMediation.ear.
 - c. Click **Next**.

Preparing for the application installation

Specify the EAR, WAR, JAR, or SAR module to upload and install.

Path to the new application

☒ Local file system

Full path
D:\IBM\WebSphere\Mon

☐ Remote file system

Full path

Context root
 Used only for standalone Web modules (.war files) and modules (.sar files)

How do you want to install the application?

☒ Prompt me only when additional information is required.

☐ Show me all installation options and parameters.

Figure 8-4 Installing the application - specify the path to the EAR file

3. In the “Select installation options” panel, click **Next**.

4. In the “Map modules to servers” panel (Figure 8-5), make sure the server selected for the AddCEIHeaderMediation module is the same as that noted in step 3 on page 216. Click **Next**.

Specify options for installing enterprise applications and modules.

[Step 1: Select installation options](#)
→ Step 2: Map modules to servers
[Step 3: Summary](#)

Map modules to servers

Specify targets such as application servers or clusters of application servers where you want to install the modules that are contained in your application. Modules can be installed on the same application server or dispersed among several application servers. Also, specify the Web targets as targets that serve as routers for requests to this application. The plug-in configuration file (plugin-cfg.xml) for each Web server is generated based on the applications that are routed through.

Clusters and Servers:
WebSphere:cell=WBMonSrv_wps_Cell,node=WBMonSrv_wps_Node,server=server1 [Apply](#)

Select	Module	URI	Server
<input type="checkbox"/>	AddCEIHeaderMediation	AddCEIHeaderMediation.jar,META-INF/ejb-jar.xml	WebSphere:cell=WBMonSrv_wps_Cell,node=WBMonSrv_wps_Node,server=server1

[Previous](#) [Next](#) [Cancel](#)

Figure 8-5 Map modules to servers panel

5. In the “Summary” panel, click **Finish**.
6. In the installation status panel (Figure 8-6), click **Save directly to the master configuration**.

Note: This panel shows a list of information and warning messages concerning the installation. As long as these are all warning or information messages, the installation has been successful

ADMA5013I: Application MQtoCEIMediation installed successfully.

Application MQtoCEIMediation installed successfully.

To start the application, first save changes to the master configuration.

Changes have been made to your local configuration. You can:

- [Save](#) directly to the master configuration.
- [Review](#) changes before saving or discarding.

To work with installed applications, click the "Manage Applications" button.

[Manage Applications](#)

Figure 8-6 Results of the application installation

7. Make sure the mediation application is started. From the administrative console, expand **Applications** and click **Enterprise Applications**. If the newly installed application is not started, select **MQtoCEIMediation** and click **Start**.

8.3.2 Running the configuration script

The second part of setting up the link between WebSphere MQ and the CEI bus is to run the configuration script provided with WebSphere Business Monitor:

1. Open a Command Prompt window. Navigate to the `<was_root>/scripts.wbm/CEIMQ` directory, as shown in Figure 8-7. Type the name of the batch file, `configCEIForMQClients.bat`, and press Enter.

```
D:\>cd IBM\WebSphere\MonServer\scripts.wbm\CEIMQ
D:\IBM\WebSphere\MonServer\scripts.wbm\CEIMQ>dir
Volume in drive D is IBM Software
Volume Serial Number is 4485-4494

Directory of D:\IBM\WebSphere\MonServer\scripts.wbm\CEIMQ

03/31/2008  10:28 PM    <DIR>          .
03/31/2008  10:28 PM    <DIR>          ..
03/31/2008  10:28 PM                563 configCEIForMQClients.bat
03/31/2008  10:28 PM            2,010 configCEIForMQClients.props
03/31/2008  10:28 PM                479 configCEIForMQClients.sh
03/31/2008  10:28 PM            4,016 MQtoCEIMediation.ear
               4 File(s)              7,068 bytes
               2 Dir(s) 19,879,370,752 bytes free

D:\IBM\WebSphere\MonServer\scripts.wbm\CEIMQ>configCEIForMQClients.bat_
```

Figure 8-7 Using the command line for running the `configCEIForMQClients` script

The `configCEIForMQClients.bat` script has two modes of execution. It can be run interactively, or it can be run using a properties file. We run it interactively.

2. Enter the following parameters when prompted:
 - a. For **Fully qualified hostname of the server hosting CEI**, press Enter to accept the default.
 - b. For **SOAP port of the server hosting CEI**, press Enter to accept the default.
 - c. For **Administrator or configurator userid**, press Enter to indicate no security enabled. The interactive script (Figure 8-8 on page 221) contains a link to documentation detailing how the connection can be secured.
 - d. To create the script instead of connecting directly to WebSphere MQ, type script and press Enter.

Note: We have not chosen to connect directly to the WebSphere MQ queue manager to configure it because we need to edit the default queue names to make them upper case due to restrictions on the use of lowercase letters for z/OS console support. See the following Web page for more information:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzac.doc/pc10510_.htm

Instead, we have chosen to generate the WebSphere MQ object definitions in an MQSC script.

- e. For the script name, press Enter to accept the default of **createMQLink.mqsc**.
- f. For **Name of the WebSphere MQ queue manager**, type MQ8G and press Enter. Ensure that the queue manager name entered here matches the queue manager defined when you added WebSphere MQ support to your CICS region ().

See 4.3.1, “Adding WebSphere MQ support to CICS region, to use the WMQ EP adapter” on page 61 for more information about adding WebSphere MQ support to you CICS region.
- g. For **Fully qualified hostname of the queue manager**, type the host name of your queue manager (in our case wtsc66.itso.ibm.com), then press Enter.
- h. For **Port to be used for communications to the queue manager**, press Enter to accept the default of port 1414.

Figure 8-8 on page 221 shows the interactive script.

This application will configure an MQLink from WebSphere MQ to CEI to allow WebSphere MQ applications to send CBEs to CEI

Enter the fully qualified hostname of the server or deployment manager hosting CEI

Hit enter to accept the default (localhost) :

Enter the SOAP port of the server or deployment manager hosting CEI :

Hit enter to accept the default (8881) :

Enter an administrator or configurator userid :

Hit enter if no security is enabled :

This configuration utility can connect directly to Websphere MQ to perform the required configuration or can produce an MQSC script that can be run manually against the queue manager

Enter 'connect' or 'script' :

script

Enter the name for the script file to be created

Hit enter to accept the default (createMQLink.mqsc) :

Enter the name of the WebSphere MQ queue manager you want to connect to CEI :

MQ8G

Enter the fully qualified hostname of the queue manager

Hit enter to accept the default (localhost) :

wtsc66.itso.ibm.com

Enter the port to be used for communications to the queue manager :

Hit enter to accept the default (1414) :

This utility will create a non secure (non-SSL) channel between CEI and WebSphere MQ. If you want the communication to be encrypted with SSL then refer to the following documentation on securing connections between WebSphere application server and WebSphere MQ.

http://www.ibm.com/developerworks/websphere/techjournal/0601_smithson/0601_smithson.html

Creating the foreign bus link

Creating an alias to the CEI destination

Adding a remote queue definition to the MQSC script

Everything completed successfully

Figure 8-8 Interactive CEI MQ script

3. When the script is completed, restart the WebSphere Business Monitor server.

All the artifacts have now been created on the WebSphere Business Monitor server and a script specifying the WebSphere MQ artefacts to be defined has been created.

The following objects were created on the WebSphere Business Monitor server:

- A foreign bus definition: MQ8G (Figure 8-9).

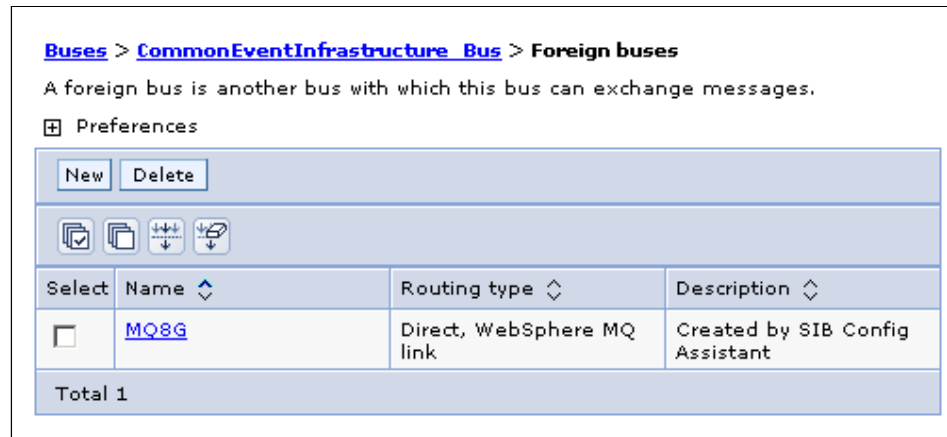


Figure 8-9 Foreign bus definition, MQ8G

- An MQ Link: Link_to_MQ8G (Figure 8-10 on page 223).

A link between the messaging engine and a WebSphere MQ network. The WebSphere MQ link connects the messaging engine as a queue manager to WebSphere MQ, thereby providing a bridge between the bus and a WebSphere MQ network.

Runtime	Configuration
<h3>General Properties</h3>	
<h4>Administration</h4>	
* Name <input type="text" value="Link_to_MQ8G"/>	
UUID <input type="text" value="199D409C9A9F1135"/>	
Description <div><input type="text" value="Created by SIB Config Assistant"/><div></div></div>	
<h4>Connection</h4>	
* Foreign bus name <input type="text" value="MQ8G"/>	
* Queue manager name <input type="text" value="CommonEventInfrastructure_"/>	
Batch size <input type="text" value="50"/>	

Additional Properties

- [Publish/subscribe broker profiles](#)
- [Receiver channel](#)
- [Sender channel](#)

Related Items

- [Foreign bus](#)

Figure 8-10 MQ Link definition, Link_to_MQ8G

- A Queue Destination (CEIQueueAliasForMQ). CEIQueueAliasForMQ is mediated using a Service Integration Bus mediation (Figure 8-12 on page 225), which adds certain properties to the inbound messages from WebSphere MQ to allow the CEI to process them. It redirects messages to the CEI server's queue, as specified in the forward routing path (Figure 8-11).

[Buses](#) > [CommonEventInfrastructure Bus](#) > [Destinations](#) > [CEIQueueAliasForMQ](#)

A queue for point-to-point messaging.

Configuration

General Properties

Identifier
CEIQueueAliasForMQ

UUID
3E2023F525B61FFB8E1975298

Type
Queue

Description

Mediation
MQ TO CEI Mediation

Quality of Service

☒ Enable producers to override default reliability

Default reliability
Assured persistent

Maximum reliability
Assured persistent

Default priority
0

Message points

- [Queue points](#)
- [Mediation points](#)

Additional Properties

- [Context properties](#)
- [Mediation execution points](#)

Figure 8-11 Alias queue destination, CEIQueueAliasForMQ

- A Service Integration Bus mediation (MQ TO CEI Mediation), which adds certain properties to the inbound messages from WebSphere MQ to allow the CEI to process them (Figure 8-12 on page 225).

Buses > CommonEventInfrastructure_Bus > Mediations

A mediation that is associated with a bus destination to apply processing to messages on that destination.

⊞ Preferences

New Delete

Select	Mediation name ↕	Handler list name ↕	Description ↕
<input type="checkbox"/>	MQ TO CEI Mediation	MQtoCEI	Created by SIB Config Assistant

Total 1

Figure 8-12 Mediation, MQ TO CEI Mediation

The script generated for creating the definitions on WebSphere MQ is shown in Figure 8-13.

```
DEFINE CHANNEL ('MQ_TO_CEI') +
  CHLTYPE (SDR) +
  CONNAME ('ibm-7iv8ob4tb6z(5559)') +
  XMITQ ('CommonEventInfrastructure_Bus') +
  DESCR ('Sender channel to SIBus') +
  NOREPLACE

DEFINE CHANNEL ('CEI_TO_MQ') +
  CHLTYPE (RCVR) +
  DESCR ('Receiver channel from SIBus') +
  NOREPLACE

DEFINE QLOCAL ('CommonEventInfrastructure_Bus') +
  INITQ('SYSTEM.CHANNEL.INITQ') +
  MAXDEPTH(200000) +
  TRIGGER +
  TRIGTYPE (EVERY) +
  TRIGDATA('MQ_TO_CEI') +
  USAGE(XMITQ) +
  DESCR ('Transmission Queue to SIBus') +
  NOREPLACE

DEFINE QREMOTE ('CEIQueue') +
  RNAME('CEIQueueAliasForMQ') +
  RQMNAME('CommonEventInfrastructure_Bus') +
  XMITQ('CommonEventInfrastructure_Bus') +
  DESCR ('Reference to the CEI Queue') +
  NOREPLACE
```

Figure 8-13 Output script (*CreateMQLink.mqsc*)

8.3.3 WebSphere MQ queue manager configuration

Because we did not choose the option to connect directly to the WebSphere MQ queue manager, the generated MQSC script needs to be run against the target WebSphere MQ queue manager (MQ8G in our example) to complete the configuration.

We are using WebSphere MQ on z/OS so we need to edit the MQSC script to make the queue names upper case as there are restrictions on the use of lowercase letters for z/OS console support. For more information about this requirement, see the following Web page:

http://publib.boulder.ibm.com/infocenter/wmqv7/v7r0/index.jsp?topic=/com.ibm.mq.csqzac.doc/pc10510_.htm

Create a file (in our case, CICSRS9.MQ.JCL(INPT1)) and set CAPS OFF. Paste in the MQ definitions that were generated in CreateMLLink.mqsc.

1. In the DEFINE CHANNEL ('MQ_TO_CEI') section add TRPTYPE (TCP).
2. Make the WebSphere MQ on z/OS queue names upper case.
 - a. Change the QLOCAL definition to use a name of CEI.XMITQUEUE, and update references to this later in the script.
 - b. Change the QREMOTE name to WBM.QUEUE.

The resultant file should look similar to Figure 8-14 on page 228.

```

DEFINE CHANNEL ('MQ_TO_CEI') +
  CHLTYPE (SDR) +
  TRPTYPE (TCP) +
  CONNAME ('ibm-7iv8ob4tb6z(5559)') +
  XMITQ ('CEI.XMITQUEUE') +
  DESCR ('Sender channel to SIBus') +
  REPLACE
*
DEFINE CHANNEL ('CEI_TO_MQ') +
  CHLTYPE (RCVR) +
  DESCR ('Receiver channel from SIBus') +
  REPLACE
*
DEFINE QLOCAL ('CEI.XMITQUEUE') +
  INITQ('SYSTEM.CHANNEL.INITQ') +
  MAXDEPTH(200000) +
  TRIGGER +
  TRIGTYPE (EVERY) +
  TRIGDATA('MQ_TO_CEI') +
  USAGE(XMITQ) +
  DESCR ('Transmission Queue to SIBus') +
  REPLACE
*
DEFINE QREMOTE ('WBM.QUEUE') +
  RNAME('CEIQueueAliasForMQ') +
  RQMNAME('CommonEventInfrastructure_Bus') +
  XMITQ('CEI.XMITQUEUE') +
  DESCR ('Reference to the CEI Queue') +
  REPLACE

```

Figure 8-14 CICSRS9.MQ.JCL(INPT1)

Create a JCL file (in our case, CICSRS9.MQ.JCL) that will read in the configuration contained in CICSRS9.MQ.JCL(INPT1). An example is shown in Figure 8-15 on page 229.

```
//CICSR91 JOB ,,NOTIFY=&SYSUID,MSGCLASS=H
/*JOBPARM S=SC66
//COMMAND EXEC PGM=CSQUTIL,PARM='MQ8G'
//STEPLIB DD DISP=SHR,DSN=MQ600.SCSQANLE
// DD DISP=SHR,DSN=MQ600.SCSQAUTH
//CSQUCMD DD DSN=CICSR9.MQ.JCL(CMDINPT1),DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
* THE NEXT STATEMENT CAUSES COMMANDS TO BE READ FROM CSQUCMD DDNAME
COMMAND
/*
```

Figure 8-15 *CICSR9.MQ.JCL (COMMAND)*

Submit the job, verify that it runs cleanly, and review the MQ resources that have been created:

1. A sender channel: MQ_TO_CEI. To display details of the sender channel, go to the WebSphere MQ for z/OS main menu and enter CHANNEL for the object type. On the next panel, type 1 alongside MQ_TO_CEI to display the channel details (Figure 8-16).

```
Display a Sender Channel - 1

Press F8 to see further fields, or Enter to refresh details.

More:      +
Channel name . . . . . MQ_TO_CEI
Disposition . . . . . : QMGR      MQ8G
Description . . . . . : Sender channel to SIBus

Transport type . . . . . : T  L=LU6.2, T=TCP/IP
Connection name . . . . . : 127.0.0.1(21000)
Local address . . . . . :
LU6.2 mode name . . . . . :
LU6.2 TP name . . . . . :

Transmission queue . . . . : CEI.XMITQUEUE
```

Figure 8-16 *MQ_TO_CEI channel*

2. A receiver channel, CEI_TO_MQ. From the list of channels, type 1 alongside CEI_TO_MQ to display the receiver channel details (Figure 8-17).

```

Display a Receiver Channel - 1

Press F8 to see further fields, or Enter to refresh details.

More:      +
Channel name . . . . . CEI_TO_MQ
Disposition . . . . . : QMGR      MQ8G
Description . . . . . : Receiver channel from SIBus

Put authority . . . . . : D  D=Default, C=Context, O=OnlyMCA,
A=AltMCA

```

Figure 8-17 CEI_TO_MQ channel

3. A remote queue definition: WBM.QUEUE targeting the alias queue in the CEI bus. This will be used to send messages to the CEI server. To display details of the remote queue definition, enter an object type of QUEUE in the WebSphere MQ for z/OS main menu. A list of queues will be displayed. Type 1 along side WBM.QUEUE to display the details of the remote queue definition (Figure 8-18 on page 230).

```

Display a Remote Queue - 1

Press F8 to see further fields, or Enter to refresh details.

Queue name . . . . . WBM.QUEUE
Disposition . . . . . : QMGR      MQ8G
Description . . . . . : Reference to the CEI Queue

Put enabled . . . . . : Y  Y=Yes, N=No
Default persistence . . . . : N  Y=Yes, N=No
Default priority . . . . . : 0  0 - 9
Remote name . . . . . : CEIQueueAliasForMQ
Remote queue manager . . . : CommonEventInfrastructure_Bus
Transmission queue . . . . : CEI.XMITQUEUE

Last alteration time . . . : 2009-07-01 10.33.29

```

Figure 8-18 WBM.QUEUE

4. A transmission queue: CEI.XMITQUEUE. The details of the transmission queue can be displayed by typing 1 alongside CEI.XMITQUEUE from the list of queues (Figure 8-19).

```
Display a Local Queue - 1

Press F8 to see further fields, or Enter to refresh details.

More:      +
Queue name . . . . . CEI.XMITQUEUE
Disposition . . . . . : QMGR      MQ8G
Description . . . . . : Transmission Queue to SIBus

Put enabled . . . . . : Y  Y=Yes, N=No
Get enabled . . . . . : Y  Y=Yes, N=No
Usage . . . . . : X  N=Normal, X=XmitQ
Storage class . . . . . : DEFAULT
CF structure name . . . . . :
Dynamic queue type . . . . . : N  N=Non-dynamic (Predefined),
T=Temporary,
                                P=Permanent, S=Shared
Page set identifier . . . . . :
Use counts - Output . . . . . : 0          Input . . . . . : 0
Current queue depth . . . . . : 0
```

Figure 8-19 CEI.XMITQUEUE

5. Restart the WebSphere Business Monitor server.

We now have a WebSphere MQ link (unsecured) between WebSphere MQ and the CEI bus. Any messages sent to the remote queue definition will be forwarded to the CEI bus for publication as an event. If the messages do not have the correct common base event format, they will be discarded.

8.3.4 Verify the configuration

In this section we will verify our configuration is correct.

Verify the MQ Link

Verify that the WebSphere MQ link has started correctly by performing the following steps:

1. From the Administrative console on the WebSphere Business Monitor server, expand **Service Integration Buses**.
2. Click **CommonEventInfrastructure_Bus**.
3. Under Topology, click **Messaging Engines**. One messaging engine should be defined and there should be a green arrow in the status field to indicate that the messaging engine is active, as shown in Figure 8-20.



Figure 8-20 Messaging engine

- Click the messaging engine. Click **Additional properties WebSphere MQ links**. One link should be defined, and there should be a green arrow in the status column, indicating that the link is active, as shown in Figure 8-21.

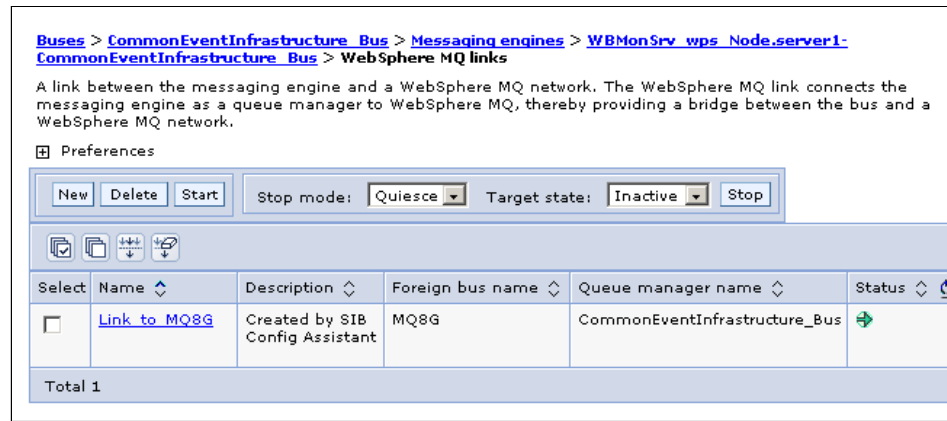


Figure 8-21 MQ Link

Verify events are received

Prior to creating the monitor model, you can also verify the configuration by emitting some events of the CBE format from the CICS application and checking that they reach the CEI server. Ensure that the ShoppingMonitorBinding is enabled in CICS, then run the application to generate some events. Use the WebSphere Business Monitor Event Management Console to check these events are received by performing the following steps:

- From the WebSphere Business Monitor Administrative Console, expand **Applications Monitor Services Recorded Events Management Events Management**, as shown in Figure 8-22.

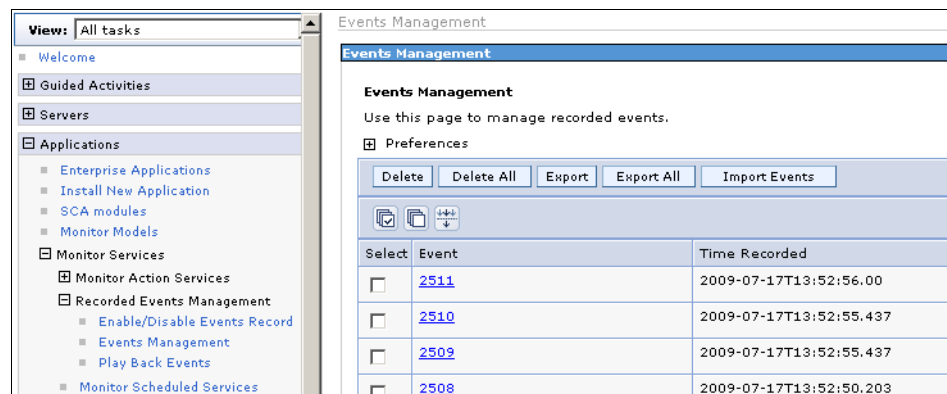


Figure 8-22 Event Management console

2. Select one of the events. You will be able to see the event payload, as shown in Figure 8-23.

[Events Management](#) > [View Event XML](#)
Use this page to view the event XML data.

Expand All

Collapse All

```
<CommonBaseEvent creationTime="2009-07-17T13:51:28.674+00:00" version="1.0.1" >
  <sourceComponentId component="IBM CICS TS#4.1.0" componentIdType="ProductName"
    executionEnvironment="IBM z/OS" instanceId="USIBMSC.EPRED5" location="SC66"
    locationType="Hostname" subComponent="CICS EP"
    componentType="http://www.ibm.com/xmlns/prod/cics/eventprocessing" />
  ☒ <situation categoryName="OtherSituation" >
    ☒ <situationType xsi:type="OtherSituation" reasoningScope="EXTERNAL" >
      <CICSApplicationEvent/>
    </situationType>
  </situation>
  ☒ <cics:event>
    ☒ <cics:context-info>
      <cics:eventname> Order</cics:eventname>
      <cics:usertag> V001</cics:usertag>
      <cics:networkapplid> USIBMSC.EPRED5</cics:networkapplid>
      <cics:timestamp> 2009-07-17T13:51:28.674+00:00</cics:timestamp>
      <cics:bindingname> ShoppingMonitorBinding</cics:bindingname>
      <cics:capturespecname> LINKPOST</cics:capturespecname>
      <cics:UOWid>
        1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F2F57F75F61C9114000200</cics:UOWid>
    </cics:context-info>
    ☒ <cics:payload-data>
      ☒ <data:payload>
        <data:CustomerNumber> +00005</data:CustomerNumber>
        <data:OrderNumber> +00020</data:OrderNumber>
        <data:StockId> +00006</data:StockId>
        <data:Quantity> +00010</data:Quantity>
      </data:payload>
    </cics:payload-data>
  </cics:event>
</CommonBaseEvent>
```

Figure 8-23 Test event payload

8.3.5 Security considerations

For the purpose of our scenario, security is turned off. For information about security on the WebSphere MQ link, refer to the Information Center.

When enabling security on the MQ Link, ensure you set the inbound user ID on the foreign bus to be a user ID that has authority to connect and send messages to the CommonEventInfrastructure_Bus, or reuse the existing one.

To set the Inbound user ID, perform the following steps:

1. From the WebSphere Business Monitor administrative console, click **Service Integration Buses**.
2. Click the **CommonEventInfrastructure_Bus**.
3. Under Topology, click **Foreign buses**. click the foreign bus that represents your WebSphere MQ queue manager.
4. Under Additional Properties, click **WebSphere MQ link routing properties**. Enter an appropriate inbound user ID and click **Apply**. Save your changes.

You can find an existing user ID that should have the necessary authority by looking at the CommonEventInfrastructureJMSAuthAlias J2C Authentication Alias:

1. From the WebSphere Business Monitor administrative console click **Security Secure administration**.
2. Under Authentication, click **Java Authentication and Authorization Service**. Then click **J2C authentication data**.
3. Look at the user ID information for the Alias CommonEventInfrastructureJMSAuthAlias. This is a user ID that should have authority to connect and send messages to the CommonEventInfrastructure_Bus.

8.4 Create monitor model

The IBM WebSphere Business Monitor development toolkit provides the tools for creating monitor models that can be transformed into executable code for WebSphere Business Monitor. The toolkit runs on WebSphere Integration Developer or Rational Application Developer, and includes the Monitor Model editor and the WebSphere Business Monitor test environment.

Our development environment for this scenario uses WebSphere Integration Developer v6201 and the WebSphere Business Monitor Toolkit v6201 on Windows 2003 Server. We have also used, as our runtime environment, the Monitor Server profile that is installed by default with the Monitor Toolkit.

Note: Instructions for installation of WebSphere Integration Developer and WebSphere Business Monitor Toolkit can be found in *z/OS: WebSphere Business Process Management V6.2 Production Topologies*, SG24-7733 and *Business Process Management Enabled by SOA*, REDP-4495.

Monitor models are XML documents that specify how information should be extracted from events at run time and collected, combined, and stored for representation on a dashboard. The completed monitor model contains a monitor details model, key performance indicator (KPI) model, dimensional model, visual model, and event model.

Tip: If you are not interested in the building of a monitor model, you can import the project from the Additional Materials and skip to 8.5, “Generate and deploy monitor model” on page 279.

The first thing needed to create the monitor model is a Business Monitoring Project.

1. Start WebSphere Integration Developer: **Start** → **Programs** → **IBM WebSphere Integration Developer** → **IBM WebSphere Integration Developer v6.2** → **WebSphere Integration Developer v6.2**.
2. Select a workspace name (we chose CICS EP).
3. Go to the Business Monitoring perspective (select Business Monitoring in the top right hand corner). See Figure 8-24.

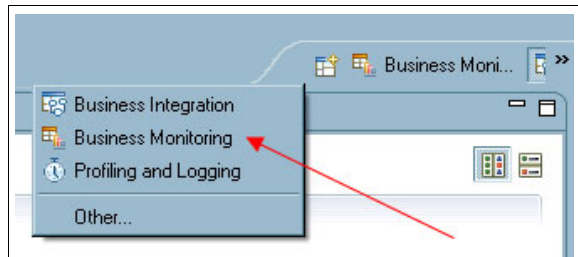


Figure 8-24 Select Business Monitoring perspective

4. Right-click in the Project Explorer, select **New Business Monitoring Project**. See Figure 8-25.

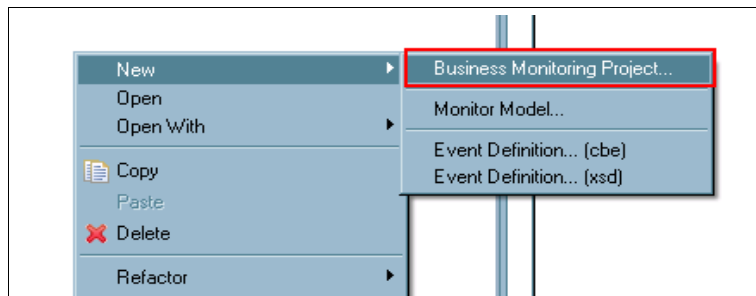


Figure 8-25 Create new project

5. Enter a name for the project (we chose RedbookMM). See Figure 8-26.

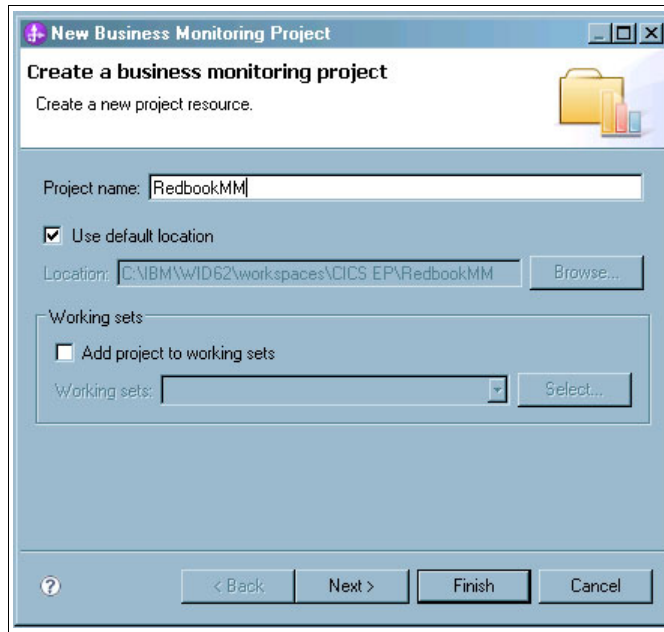


Figure 8-26 Create new project

6. Click **Finish**.

8.4.1 Import event definition schemas

We now need to import the schemas that define the structure of the events to be received.

The Business Monitor will receive the events in CBE (Common Base Event) format. The CBE is comprised of two portions:

- ▶ The CBE envelope, which consists of standard CBE elements
- ▶ The xs:any slot, which contains the application data (or payload).

In the case of the CICS events, the payload consists of two parts, context and environment information (known as the static schema portion) and information that is specific to the event (the business information, which is known as the dynamic schema portion).

Consequently, there are two types of schema we need to import. The first of these is for the static schema.

1. Right-click the Project Explorer and select **Import**.
2. Navigate to the directory where the schema file (`cicsStatic.xsd`) is located.
3. Select the `cicsStatic.xsd` file.
4. Check that the import folder is our RedbookMM project.
5. Click **Finish**.

Note: The static schema is shipped with CICS and is available on zFS with the other supplied schemas (such as event binding schemas). All the schemas used in the Business Monitor scenario are supplied in the Additional Materials.

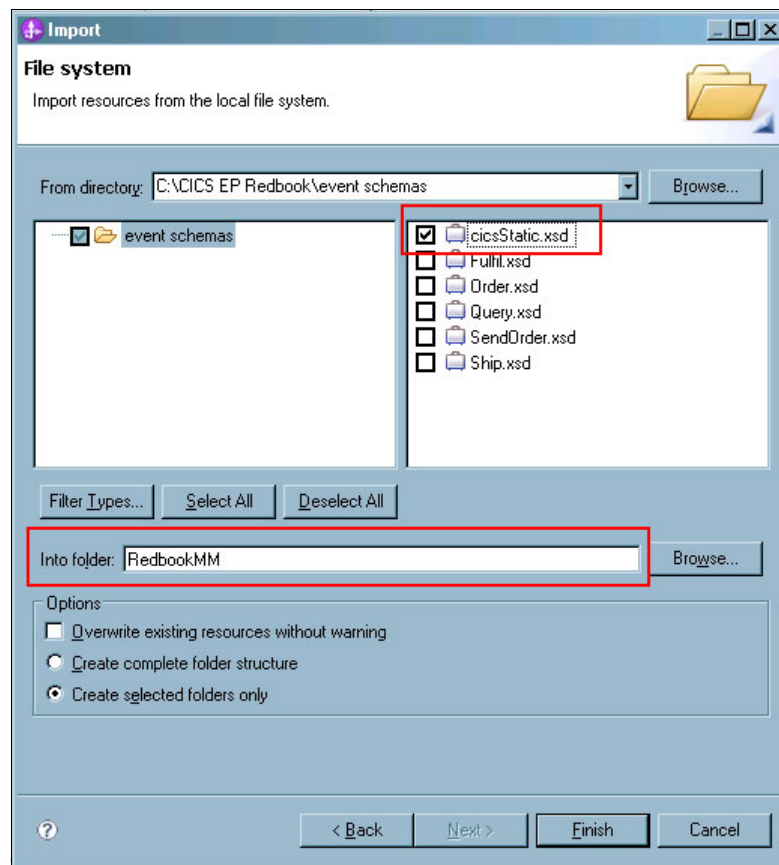


Figure 8-27 Import schema

We now need to import the schemas that define the dynamic parts of the events ().

6. Repeat the import steps and select the following schemas:
 - Order.xsd
 - QueryStock.xsd
 - SendOrder.xsd
 - Ship.xsd
7. Expand the Event Definitions folder in the Monitor project to see the list of event definitions.

Note: The additional event (ActionServicesEvent.cbe) that is seen in Figure 8-28 on page 239 is added when you create a monitoring project. It is used for outbound events. We will not be using outbound events in our scenario.

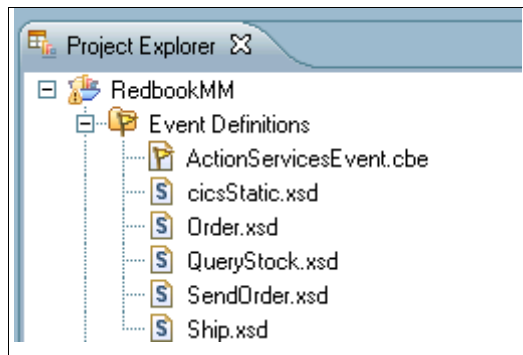


Figure 8-28 Events

We can open these schema files and see the event definitions that have been generated from CICS.

Important: You may recall that some of the fields in the shopping application are numeric. Note in Example 8-1 on page 240 that all of the event data is defined in the schema as string type. We need to ensure that the metrics in our monitor model recast this event data to the data types that we would like to be used in our model and our dashboards.

Example 8-1 SendOrder schema

```
<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://www.ibm.com/prod/cics/V001/SendOrder"
xmlns:tns="http://www.ibm.com/prod/cics/V001/SendOrder"
elementFormDefault="qualified" attributeFormDefault="qualified"
xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="payload">
    <complexType>
      <sequence>
        <element type="string" name="CustomerNumber"/>
        <element type="string" name="OrderNumber"/>
        <element type="string" name="CustomerName"/>
        <element type="string" name="City"/>
        <element type="string" name="Country"/>
        <element type="string" name="Premium"/>
        <element type="string" name="TotalOrderValue"/>
      </sequence>
    </complexType>
  </element>
</schema>
```

Now that we have the event definitions. We can now move on to creating the monitor details model.

8.4.2 Creating the monitor details model

The monitor details model is a container for monitoring contexts and their associated metrics, keys, counters, stopwatches, triggers, and events. The monitor details model holds most of the monitor model information.

1. Right-click the RedbookMM project and select **New Monitor Model**. See Figure 8-29.
2. Select a name for the monitor model (we chose RedbookMM).
3. Click **Finish**.

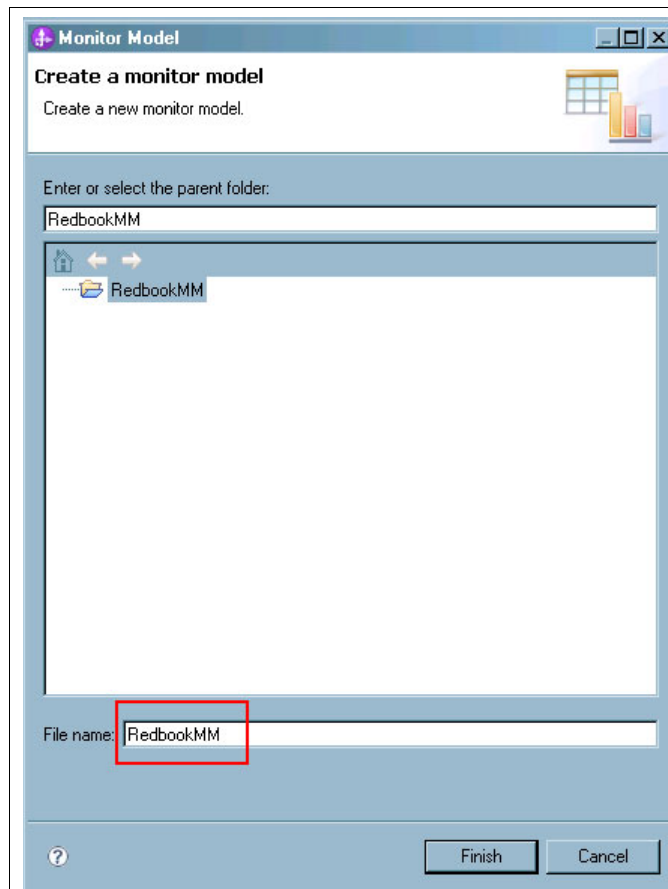


Figure 8-29 Create monitor model

The new monitor model will be opened.

Note: There will be errors at this point, but do not worry, these will be cleared up as we go.

We can see in the monitor model something called RedbookMM MC. See Figure 8-30. This is a monitoring context. A monitoring context definition defines all of the data that should be collected about an entity (such as a process, customer, order and so on) as the system is running. Each of its runtime instances (monitoring context instances) use incoming events to monitor a particular entity. As we described before, the main entity that we want to monitor (and subsequently analyze) is the customer.

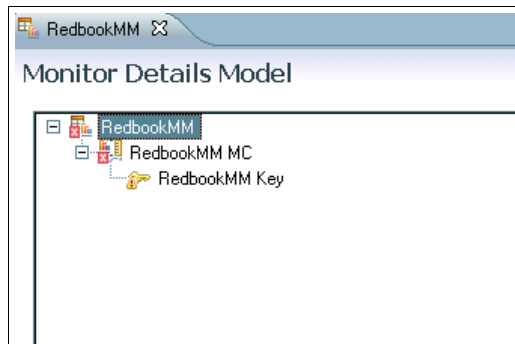


Figure 8-30 New monitor details model

The key that uniquely identifies a monitoring context instance is based on the piece or pieces of information that characterize what you want to monitor. For our example, the monitoring context represents customer activity, its key corresponds to the customer number.

A new monitor model has a default monitoring context and key created. We are going to delete these and create one with a more meaningful name.

Note: You could just as easily rename them instead.

1. Right-click the monitoring context name and select **Delete**.
2. Right-click the monitor model name and select **New Monitoring Context**.
See Figure 8-31 on page 243.
3. Give this context a name (we chose Customer MC).
4. Click **OK**.

Monitoring Context Details
Edit the details of the monitoring context, which contains the information for monitoring.

ID:

Name:

Description:

Figure 8-31 New monitoring context

Define events

We need to create an inbound event for each event definition that corresponds to an event that we want to monitor (that is, each event definition that we imported in the previous steps). We then refine the monitor model to extract the relevant data from the events as they arrive at run time. For the top-level monitoring context, the events that we are interested in are those for the customer order and the stock query. We first define the events that relate to the orders (there are two of them).

1. Select **New Inbound Event**.
2. Give the event a name (we chose OrderEvent).

We also need to create a new key. As we mentioned earlier, the key will uniquely define each monitoring context instance (in our case - each different customer).

3. Right-click the monitoring context (Customer MC) and select **New Key**.
4. Name the key Customer Number.

Key Details
Edit the details of the key. Each monitoring context requires at least one key.

ID:

Name:

Description:

Type:

Figure 8-32 Create key

Note: You will note that when you create object names with a space in them, the ID will have an underscore. No spaces or special characters are allowed in these IDs, but the toolkit will take care of that for you.

We now define the event parts. An event part is an XML schema definition (XSD) type that provides information about the structure of part of an event. A single event definition can have different event parts that are defined by different XML schemas. All the event parts, together with the CBE definition (if one is provided) describe the structure of the entire event. As mentioned earlier, events from CICS are comprised of two parts (the static and the dynamic).

We create an event part to address the cicsStatic schema, as follows.

5. In the Event Part Details portion of the panel, select **Add**. See Figure 8-33.

The screenshot shows a software interface titled "Event Type Details". Below the title is a text area with instructions: "Specify the event type or the XML schemas that together describe the structure of this inbound event. You can specify an extension name, event parts, or both." Below this is a text input field labeled "Extension name:" followed by "Browse..." and "Clear" buttons. Underneath is a section labeled "Event parts:" followed by a table with four columns: "ID", "Name", "Type", and "Path". The table is currently empty. At the bottom right of the panel, there are two buttons: "Add" and "Remove". The "Add" button is highlighted with a red rectangular box.

Figure 8-33 Create event part

6. Give the event part a name (we chose cics:Event). See Figure 8-34 on page 245.
7. Click **Select Type**, to define the data type for this event part.

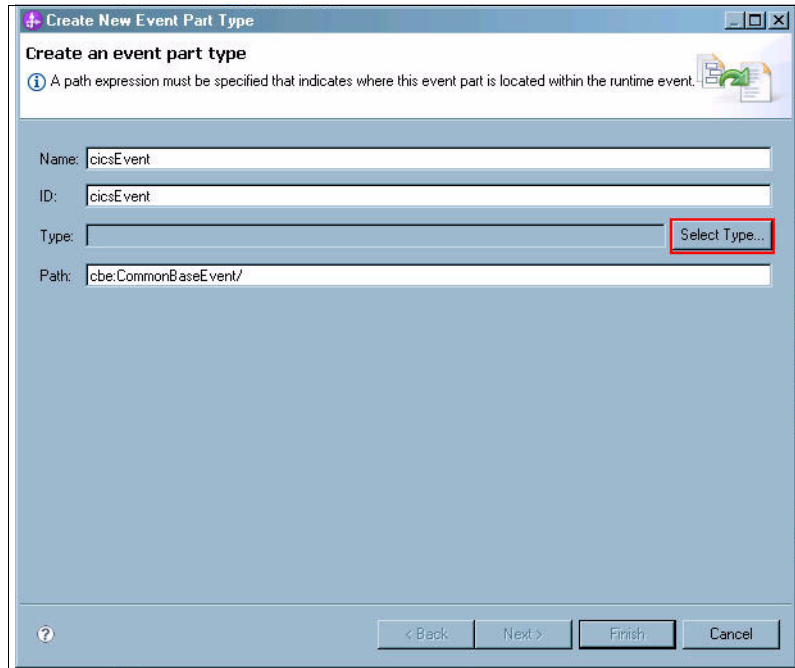


Figure 8-34 Event name

8. A list will display, allowing you to select the data type. This is where we will be utilizing the schemas.
9. Select the **Choose the data type** radio button.
10. Expand the `cicsStatix.xsd` entry.
11. Select the `cics:event` element definition. See Figure 8-46 on page 255.
12. Click **Next**.

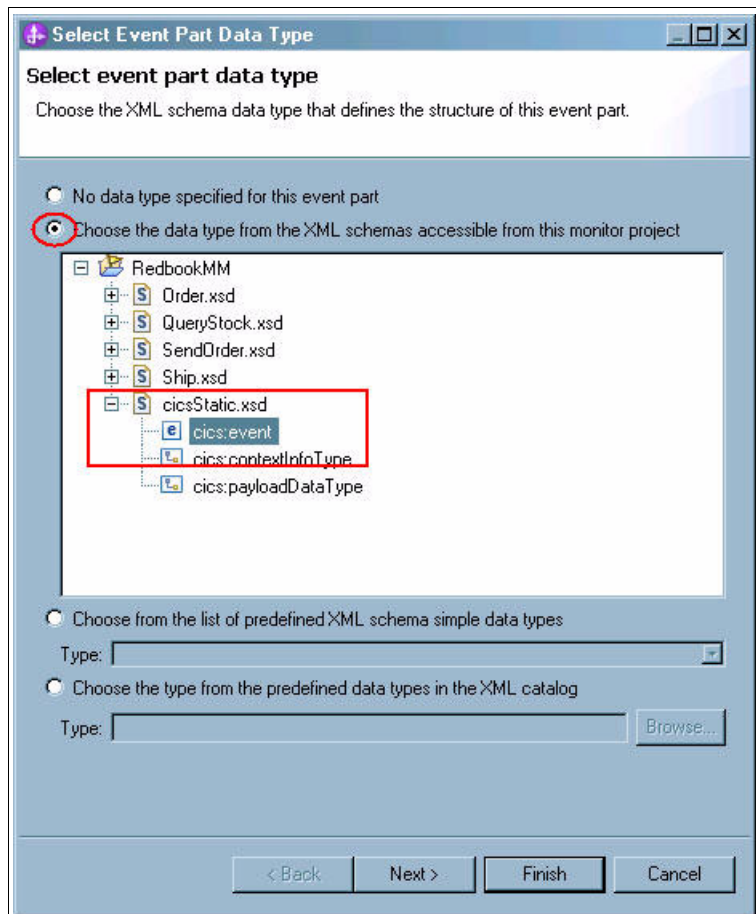
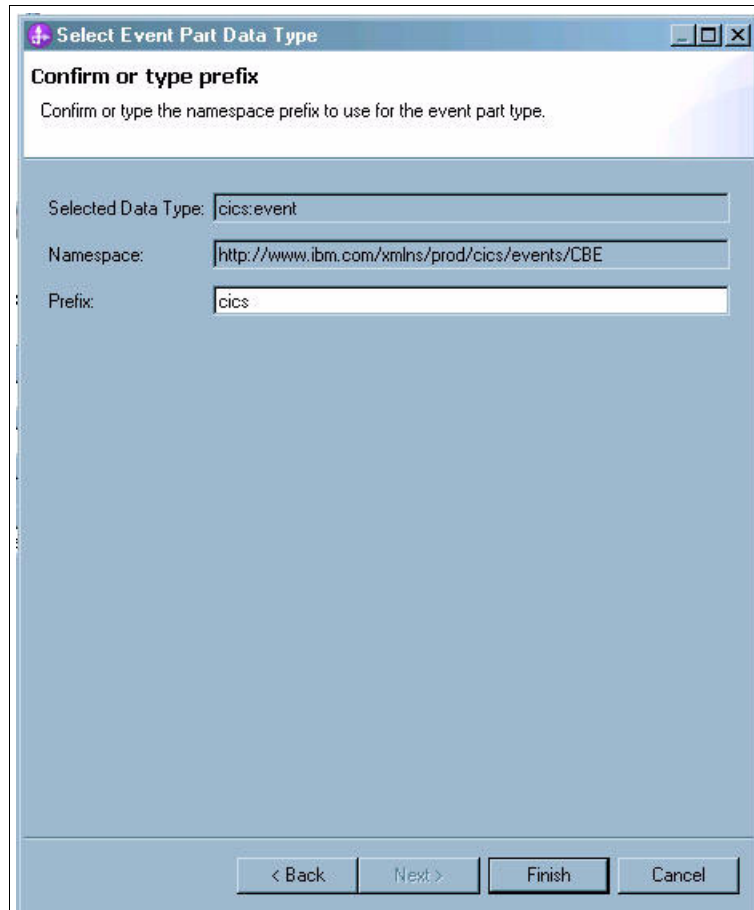


Figure 8-35 Select cics:event data type

13. On the next panel you will now see that the namespace for this schema has been inserted and a suggested namespace prefix.
14. Confirm the prefix (cics, in this case, is a good one so we used it). See Figure 8-36.
15. Click **Finish**.



Select Event Part Data Type

Confirm or type prefix
Confirm or type the namespace prefix to use for the event part type.

Selected Data Type:

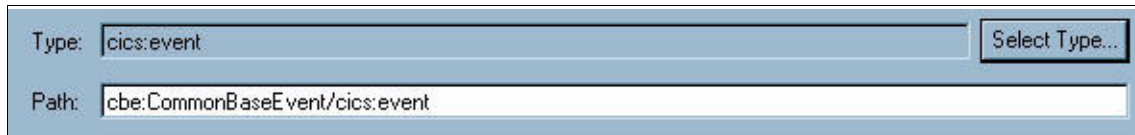
Namespace:

Prefix:

< Back Next > Finish Cancel

Figure 8-36 Confirm namespace prefix

16. You will now notice that the event part definition now contains the (xpath) path to the cics:event. See Figure 8-37 on page 248.
17. Click **Finish**.



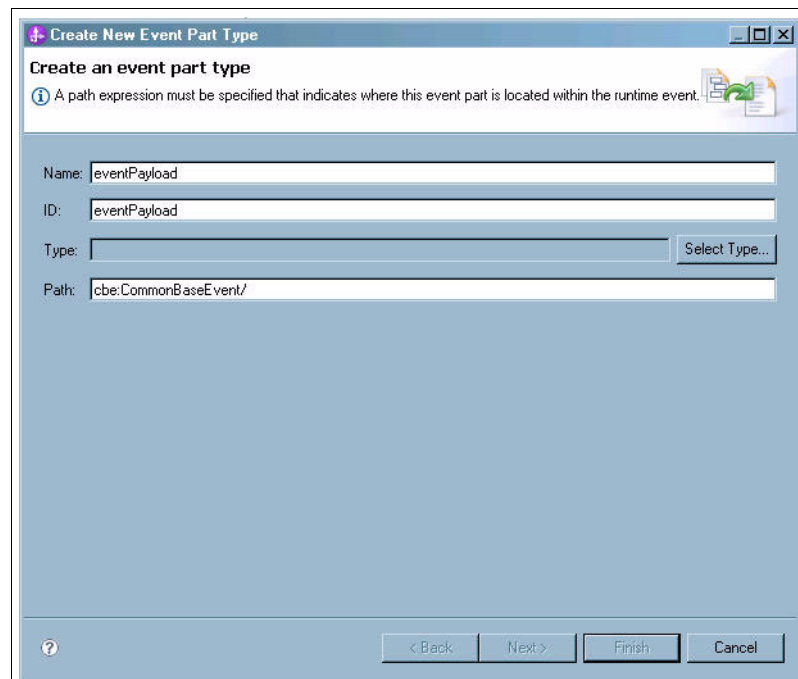
A screenshot of a form with two text input fields. The first field is labeled 'Type:' and contains the text 'cics:event'. To its right is a button labeled 'Select Type...'. The second field is labeled 'Path:' and contains the text 'cbe:CommonBaseEvent/cics:event'.

Figure 8-37 event part with path

We now need to create a second event part that addresses the event payload (that is, the business data).

18. Create a new event part (we named ours event:Payload). See Figure 8-38

19. Select the type.



A screenshot of a 'Create New Event Part Type' dialog box. The title bar says 'Create New Event Part Type'. Inside, there's a section 'Create an event part type' with an information icon and a note: 'A path expression must be specified that indicates where this event part is located within the runtime event.' Below this are four input fields: 'Name:' with 'eventPayload', 'ID:' with 'eventPayload', 'Type:' (empty) with a 'Select Type...' button, and 'Path:' with 'cbe:CommonBaseEvent/'. At the bottom are buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

Figure 8-38 Event part - event:payload

20. The type for the payload in the order event is **order:payload**. Select it. See Figure 8-39.
21. Click **Next**.

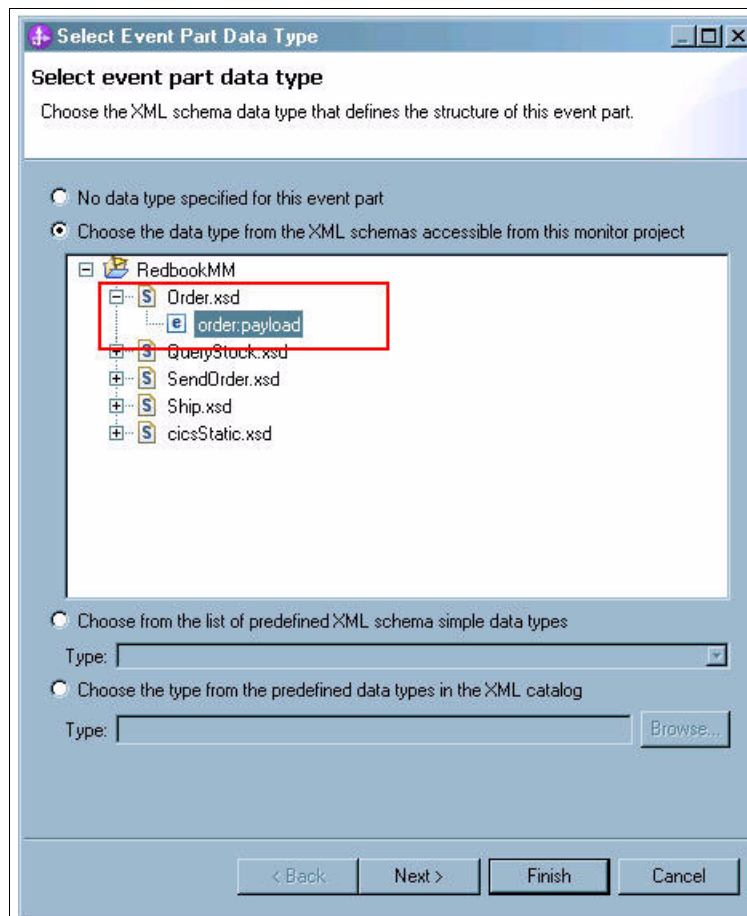
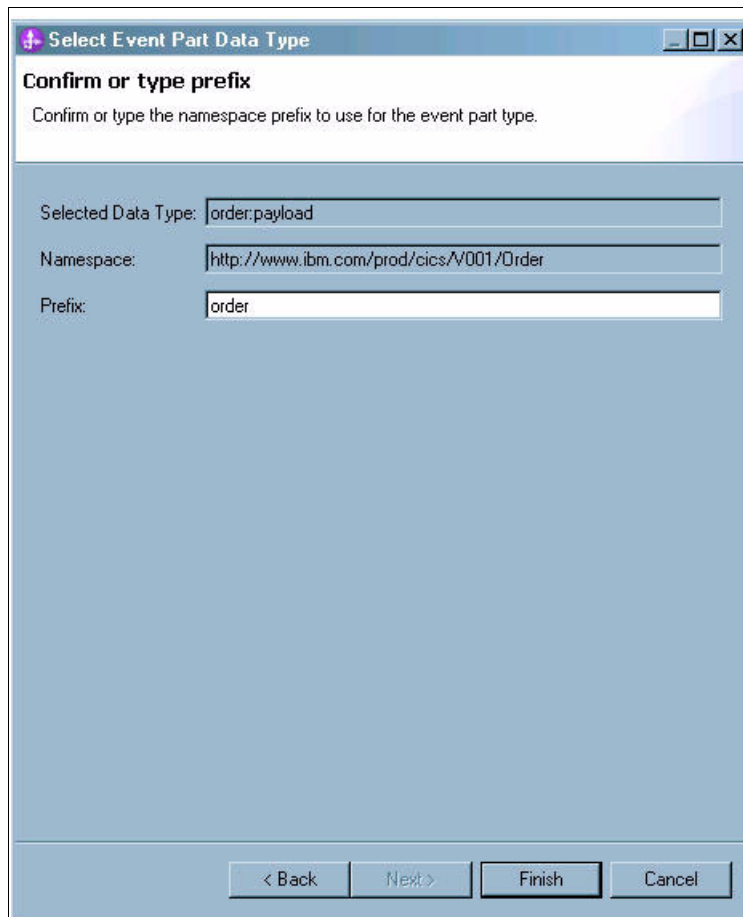


Figure 8-39 Select data type

22. Confirm the namespace prefix for this event part (we chose **order**). See Figure 8-40 on page 250.
23. Click **Next**.



The image shows a Windows-style dialog box titled "Select Event Part Data Type". It has a standard title bar with minimize, maximize, and close buttons. The main content area is light blue and contains the following elements:

- Section Header:** "Confirm or type prefix" in bold black text.
- Instruction:** "Confirm or type the namespace prefix to use for the event part type." in a smaller black font.
- Form Fields:**
 - "Selected Data Type:" followed by a text box containing "order:payload".
 - "Namespace:" followed by a text box containing "http://www.ibm.com/prod/cics/V001/Order".
 - "Prefix:" followed by a text box containing "order".
- Buttons:** At the bottom, there are four buttons: "< Back", "Next >", "Finish", and "Cancel".

Figure 8-40 order namespace

24. Specify the path to the event payload (as mentioned before, this is xpath).

cbe:CommonBaseEvent/cics:event/cics:payload-data/order:payload

See Figure 8-41.

Note: This will normally be automatically done for you, but if not, you can simply type in the required xpath expression.

25. Click **Finish**.

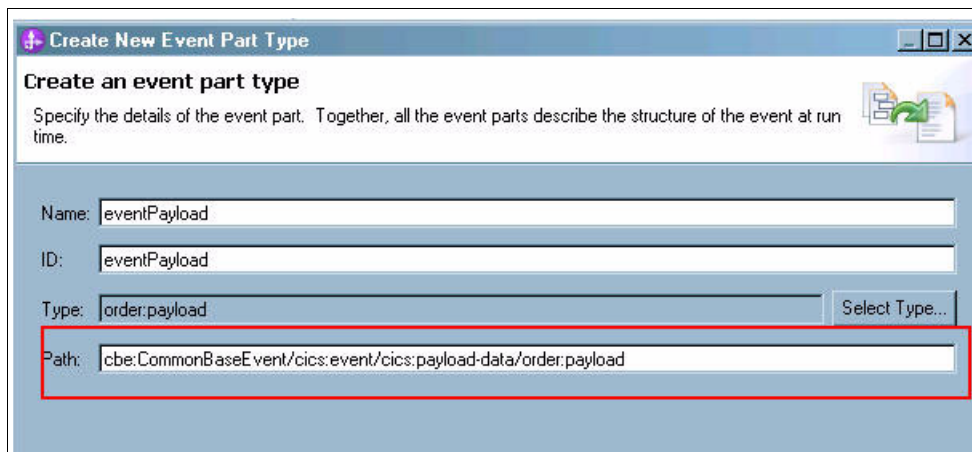
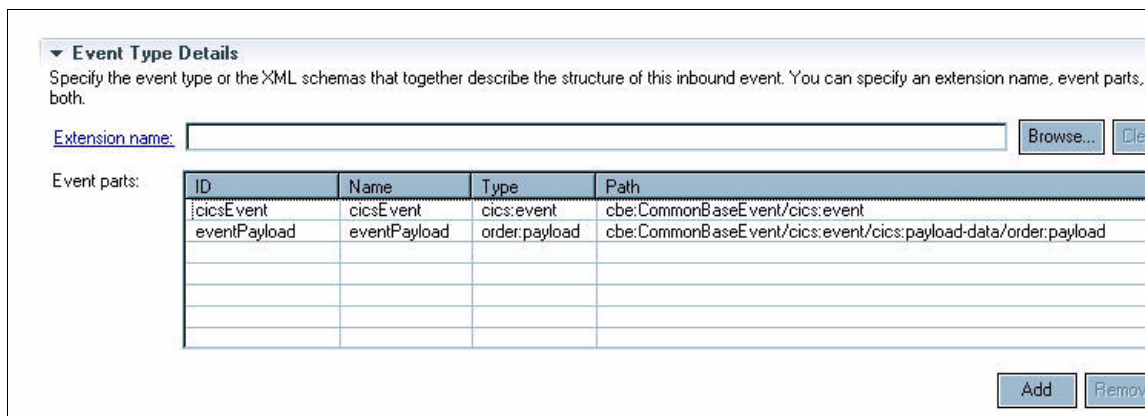


Figure 8-41 Path to event payload

Figure 8-42 shows the complete event parts.



ID	Name	Type	Path
cicsEvent	cicsEvent	cics:event	cbe:CommonBaseEvent/cics:event
eventPayload	eventPayload	order:payload	cbe:CommonBaseEvent/cics:event/cics:payload-data/order:payload

Figure 8-42 Completed event parts

Using the same method, create inbound events, as follows:

26. SendOrder (as shown in Figure 8-43)

- name: SendOrderEvent
- namespace prefix for payload: send
- data element: payload from SendOrder.xsd

27. Query (as shown in Figure 8-44 on page 253)

- name: QueryEvent
- namespace prefix for payload: query
- data element: payload from Query.xsd

The screenshot shows two configuration windows for an inbound event. The top window, titled 'Inbound Event Details', contains fields for ID (SendOrderEvent), Name (SendOrderEvent), and a Description field. The bottom window, titled 'Event Type Details', contains an 'Extension name' field and a table for 'Event parts'.

Inbound Event Details
Edit the details of the inbound event, which references an event that is generated by the monitored application.

ID:

Name:

Description:

Event Type Details
Specify the event type or the XML schemas that together describe the structure of this inbound event. You can specify an extension name, event parts, or both.

Extension name:

Event parts:

ID	Name	Type	Path
cicsEvent	cicsEvent	cics:event	cbe:CommonBaseEvent/cics:event
eventPayload	eventPayload	send:payload	cbe:CommonBaseEvent/cics:event/cics:payload-data/send:payload

Figure 8-43 SendOrder event

[-] [e] CommonBaseEvent	
[a] creationTime	2009-07-16T09:24:53.553+00:00
[a] version	1.0.1
[+] [e] sourceComponentId	
[+] [e] situation	
[-] [e] cics:event	
[a] xmlns:cics	http://www.ibm.com/xmlns/prod/cics/events/CBE
[-] [e] cics:context-info	
[e] cics:eventname	Order
[e] cics:usertag	V001
[e] cics:networkapplid	USIBMSC.EPRED4
[e] cics:timestamp	2009-07-16T09:24:53.553+00:00
[e] cics:bindingname	ShoppingMonitorBinding
[e] cics:capturespecname	LINKPOST
[e] cics:UOWid	1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F1F87DF879777584000200
[-] [e] cics:payload-data	
[-] [e] data:payload	
[a] xmlns:data	http://www.ibm.com/prod/cics/V001/Order
[e] data:CustomerNumber	+00005
[e] data:OrderNumber	+00028
[e] data:StockId	+00006
[e] data:Quantity	+00010

Figure 8-45 Example of order event sent by CICS

1. Select the OrderEvent definition.
2. Place your cursor in the editing pane and press Ctrl + Space to bring up the Content Assist. See Figure 8-46.
3. Navigate down the tree of the OrderEvent definition to locate the eventname (in the cics:event /context:info).

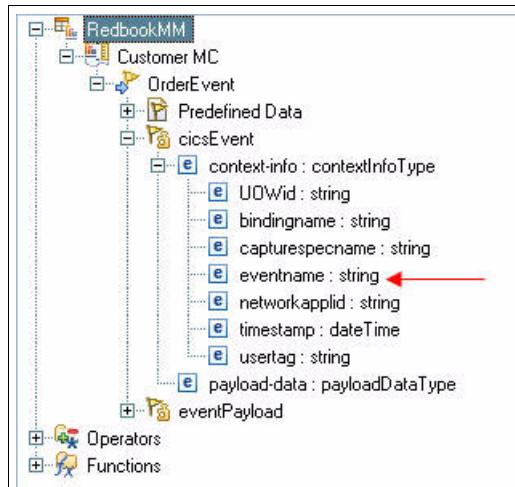


Figure 8-46 Content assist

4. Double-click this element to bring it in to the editing panel.
5. Type in the rest of the expression (= **'Order'**) to complete the filter condition. See Figure 8-47.

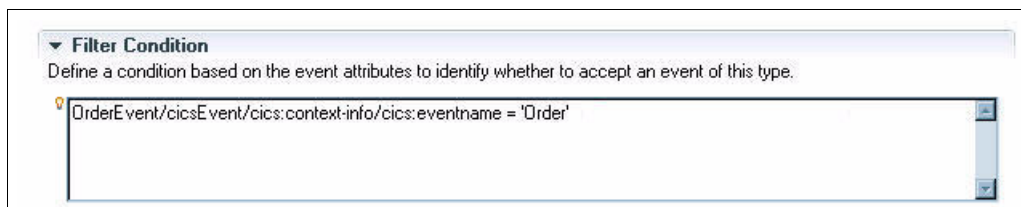


Figure 8-47 Order event filter

We next set the correlation expression. You may have noticed in Figure 8-45 on page 254, that the event payload has plus (+) signs. You may also recall that we mentioned that the schema definitions type all elements as string. The + sign indicates that this element has come from a field that is defined as numeric to the CICS application. The customer number in that figure shows

as “+00005”. We know from our application that the customer number is actually “00005,” so we need to remove the + sign to ensure that our data does not look incorrect.

Note: As the person building a monitor model, it is always a good idea to have a discussion with the people who really understand the applications that are sending the events, to ensure that everyone has the same idea of what the event data really means and how it should look.

The way that our correlation works is to check whether any monitoring context instances are interested in this event. In our case, does this event have the same customer number as the one that this instance is monitoring? If you recall that the Customer_Number is our key (the correlation expression is checking whether the CustomerNumber in the event is the same) if so, then we are interested.

6. Using the Content Assist, select the Customer_Number (double-click to enter in to the expression).
7. Enter = (equal sign) into the expression.
8. Using Content Assist, locate the xpath Functions library folder. See Figure 8-48.

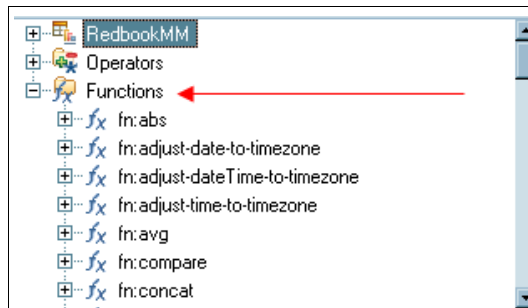


Figure 8-48 xpath functions

9. Navigate to the fn:substring function and double-click to enter in to the expression.
10. Using Content Assist, navigate to the CustomerNumber element in the event payload and double-click to enter into the expression.
11. Complete the expression by entering ,2 after the element name (we want to start from the second character of the string).

12. The expression should look as follows:

```
Customer_Number =  
fn:substring(OrderEvent/eventPayload/order:CustomerNumber,2)
```

13. After the expression has been evaluated, we need to instruct the Monitor on what to do when the expression evaluates to true.

Set as shown in Figure 8-49.



If no instances are found	Create new instance
If one instance is found	Deliver to the instance
If multiple instances are found	Treat as error

Figure 8-49 Event delivery settings

We have set this event as follows:

- If no instances are found (that is, we have not seen anything related to this customer before), create a monitoring context instance.
- If one instance is found (that is, we have seen this customer before and we are receiving new or additional events about them), deliver to the instance.
- If multiple instances are found (this should never happen), treat as an error.

14. Using the same method as above, create the filter conditions, correlation expressions, and event delivery settings for the SendOrder event and the Query event. See Figure 8-50 on page 258.

Important: Be careful to ensure that you are pointing at the correct events and event parts in your expressions.

The event delivery settings can be the same for all events, as we do not care which of these events comes first (a customer may query an item first or order an item first, we are not concerned which). See Figure 8-51 on page 258. Whichever is the first event creates the monitoring context instance for the customer, and whoever comes next provides additional information about the activities of this customer.

▼ **Filter Condition**

Define a condition based on the event attributes to identify whether to accept an event of this type.

SendOrderEvent/cicsEvent/cics:context-info/cics:eventname = 'SendOrder'

▼ **Correlation Expression**

Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

Customer_Number = fn:substring(SendOrderEvent/eventPayload/send:CustomerNumber,2)

If no instances are found
Create new instance

If one instance is found
Deliver to the instance

If multiple instances are found
Treat as error

Figure 8-50 SendOrder event

▼ **Filter Condition**

Define a condition based on the event attributes to identify whether to accept an event of this type.

QueryEvent/cicsEvent/cics:context-info/cics:eventname = 'QueryStock'

▼ **Correlation Expression**

Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

Customer_Number = fn:substring(QueryEvent/eventPayload/query:CustomerNumber,2)

If no instances are found
Create new instance

If one instance is found
Deliver to the instance

If multiple instances are found
Treat as error

Figure 8-51 Query event

The last thing we need to do is set the key. The first event that creates the context instance will set the key (this is a one off thing).

15. Select the Key (Customer Number).
16. Using the Content Assist, add an expression for the CustomerNumber in each of the inbound events (as we mentioned, it may be any one of these events that sets the key). See Figure 8-52.

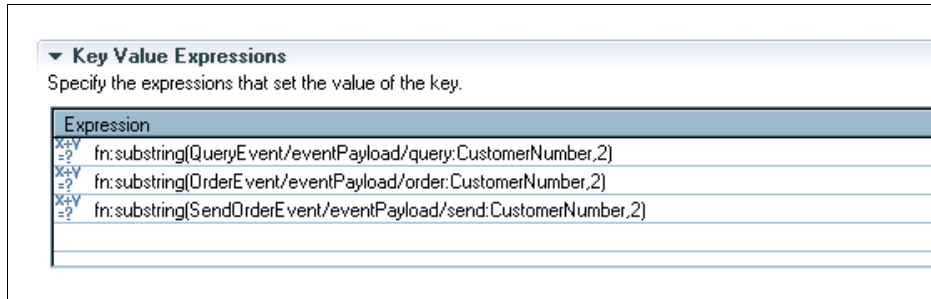


Figure 8-52 Key value expression

17. Use Ctrl + S to save any work so far.

Create metrics

A metric is a holder for information, usually business data or performance measurement, in a monitoring context. Every metric is associated with one or more expressions that define how the metric gets its value.

At the customer level of monitoring we define the following metrics:

- ▶ Customer name
 - ▶ Whether they are a premium customer
 - ▶ Counters for the number of orders / queries for that customer.
1. Right-click the monitoring context and select **New Metric**. Select a name for the metric (we chose Customer Name).
 2. Accept the default type of string.
 3. Click **OK**.
 4. Click **Add** and use the Content Assist to set the value of the metric from the CustomerName element in the SendOrderEvent. See Figure 8-53 on page 260.

▼ Metric Details

Edit the details of the metric, which is a holding spot for information used in other calculations.

ID:

*Customer_Name

Edit...

Name:

Customer Name

Description:

Type:

*String

Maximum String Length:

256

☐ Allocate additional space in database to accommodate Unicode string for globalization

☐ A value is required for this metric

Default Value:

Edit...

☒ This metric can be used for sorting

▼ Metric Value Expressions

Specify the expressions that set the value of the metric. If a trigger is specified, the expression is evaluated when the trigger fires.



Trigger	Expression
	 SendOrderEvent/eventPayload/send:CustomerName

Figure 8-53 Customer Name metric

- Repeat the steps to create a metric called Premium Customer whose value is set from the Premium element of the SendOrderEvent. See Figure 8-54.

▼ Metric Details
Edit the details of the metric, which is a holding spot for information used in other calculations.

ID:

Name:

Description:

Type:

Maximum String Length:

☐ Allocate additional space in database to accommodate Unicode string for globalization

☒ A value is required for this metric

Default Value:

☐ This metric can be used for sorting

▼ Metric Value Expressions
Specify the expressions that set the value of the metric. If a trigger is specified, the expression is evaluated when the trigger fires.



Trigger	Expression
	 SendOrderEvent/eventPayload/send:Premium

Figure 8-54 Premium metric

A counter is a special kind of metric that can be incremented, decremented, or reset to zero based on the arrival of events (or triggers).

- Select **New Counter**.
- Name the counter (we chose Customer Orders Received).
- Click **Add** to define what causes actions on this counter.
- Select **OrderEvent** as the event and **Add One** as the action.

Each time an OrderEvent is received for this customer the counter will be incremented See Figure 8-55 on page 262.

▼ Counter Details

Edit the details of the counter, which counts the number of occurrences of some situation or event.

ID: *Customer_Orders_Received Edit...

Name: Customer Orders Received

Description:

☐ This counter can be used for sorting

▼ Counter Controls

Specify what causes the counter to change and what action is taken.

Trigger / Inbound Event	Resulting Action
OrderEvent	Add One

Add Remove

Figure 8-55 Order counter

- Perform the same for a counter named Customer Queries, which is incremented each time a QueryEvent arrives for this customer.

Create child monitoring contexts

Monitoring context definitions can be nested to form hierarchies. In our example, we want to track the orders placed for each customer, but we also want to track the queries that customer has made. Given that there may be many of each of these tasks for the customer, we monitor these activities using a nested monitoring context to track each instance of the task separately.

Tip: The use of child monitoring contexts allows us to drill-down from the parent to the child in the dashboards. In our example, the parent context is the customer. We use child contexts to allow us to drill-down to see the following information:

- Orders for the customer
- Queries for customer.

At run time, each instance of a nested monitoring context definition (or child context) must be subordinate to an instance of the nesting monitoring context definition (its parent context). In other words, the instances of nested monitoring context definitions form a tree structure.

Figure 8-56 shows the hierarchy of monitoring contexts. We have created a child context for queries and one for orders.

Complete the following child contexts and their contents (or import the supplied monitor model Project Interchange™ file from the Additional Materials).

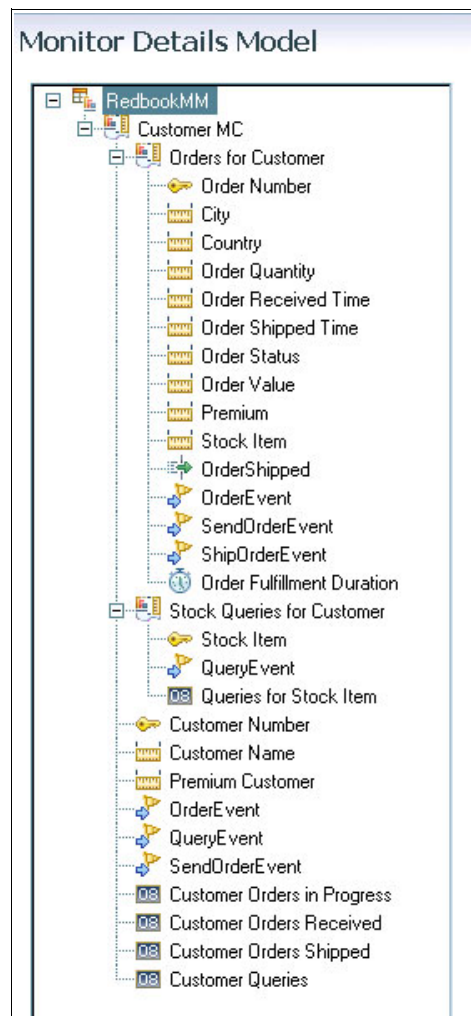


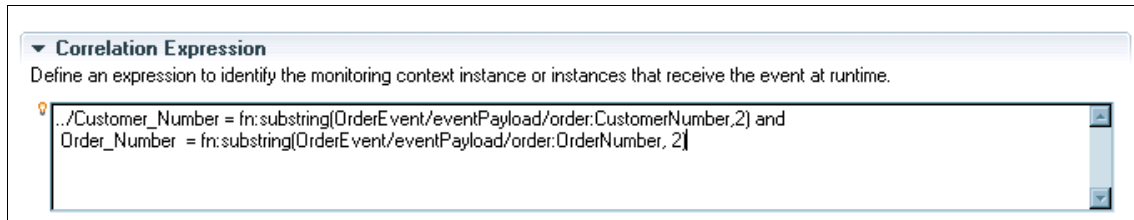
Figure 8-56 Completed monitor details model

Context: Orders for Customer

In this section we will look at orders for customers. See “Create child monitoring contexts” on page 262 for more information.

1. Events

These events are essentially the same as the events from the parent (ShipOrderEvent is a new event). The major difference is that the correlation expression must check for the key value of this context, but also check for a matching parent context. For example, we know which order the information relates to, but which customer monitoring context does it belong to? See Figure 8-57.



The screenshot shows a configuration window titled "Correlation Expression". Below the title is a description: "Define an expression to identify the monitoring context instance or instances that receive the event at runtime." Below this is a text area containing the following expression: `../Customer_Number = fn:substring(OrderEvent/eventPayload/order:CustomerNumber,2) and Order_Number = fn:substring(OrderEvent/eventPayload/order:OrderNumber, 2]`. There is a small orange lightbulb icon to the left of the text area.


Figure 8-57 Child correlation expression

As we see from the correlation expression (Figure 8-57) it is essential that the children correlate with their parent monitor context. Create these event definitions and ensure correct correlation.

- OrderEvent
- SendOrderEvent
- ShipOrderEvent

The ShipOrderEvent has different event delivery settings (Figure 8-58) as you would not expect to see an order shipping if there was no order. Therefore we set the event delivery to error if a ShipOrderEvent arrives but we have no monitor context instance for that order.

Note: For the purposes of example, we could simply ignore this event (this is an option), however in a true business scenario you would probably not want to do this. The monitor might be the one to detect that something is amiss.



The screenshot shows a configuration window for event delivery. It contains three rows, each with a label and a dropdown menu:

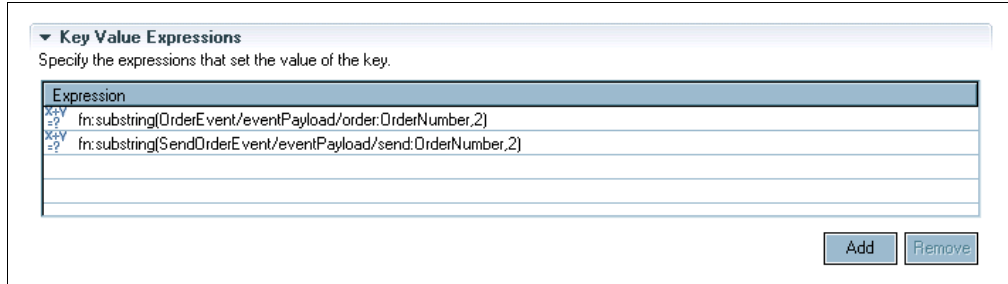
If no instances are found	Treat as error
If one instance is found	Deliver to the instance
If multiple instances are found	Treat as error

Figure 8-58 ShipOrder event delivery



2. Set key (Figure 8-59)

► Order Number

As before, either of the order events can create the context (but the ShipOrderEvent does not).



▼ Key Value Expressions
Specify the expressions that set the value of the key.

Expression
 fn:substring(OrderEvent/eventPayload/order:OrderNumber,2)
 fn:substring(SendOrderEvent/eventPayload/send:OrderNumber,2)

Add Remove

Figure 8-59 Set order key

3. Metrics (Table 8-1)

Use the content assist to find the event elements that will be used to update these metrics. Remember that some of these metrics will need to be recast from string to either integer (such as quantity) or decimal (such as value).

Table 8-1 Metrics

Metric	Map Expression®
City	SendOrderEvent/eventPayload/send:City
Country	SendOrderEvent/eventPayload/send:Country
Order Quantity	xs:integer(OrderEvent/eventPayload/order:Quantity)
Order Received Time	(1) if (fn:exists(Order_Received)) then (Order_Received) else (OrderEvent/predefinedData/creationTime) (2) if (fn:exists(Order_Received)) then (Order_Received) else (SendOrderEvent/predefinedData/creationTime) (We are using two expression here as it may be either the order event or the sendOrder event that arrives first, whichever arrives first should be the one to populate the received time metric).
Order Shipped Time	ShipOrderEvent/predefinedData/creationTime
Order Status	(1)if (fn:exists(OrderEvent/eventPayload/order:OrderNumber)) then 'IN PROGRESS' else Order_Status (2)if (fn:exists(ShipOrderEvent/eventPayload/ship:OrderNumber)) then 'SHIPPED' else Order_Status

5. Stopwatch

Stopwatches are used to measure the time between two events. Stopwatches can be started, stopped, and reset by triggers or by the arrival of inbound events. In our example, we want to measure the time it takes for the order to be shipped.

- Create a stopwatch named Order fulfillment Duration, this stopwatch can be started by either of our “creation events” (Order or SendOrder) and is stopped by the arrival of the ShipOrderEvent.

The screenshot shows a configuration window for a stopwatch. The 'Stopwatch Details' section includes fields for ID (OrderFulfillmentDuration), Name (Order Fulfillment Duration), and a description box. Below these are two checkboxes: 'This stopwatch is an accumulating stopwatch' and 'This stopwatch can be used for sorting'. The 'Stopwatch Controls' section contains a table with two columns: 'Trigger / Inbound Event' and 'Resulting Action'.

Trigger / Inbound Event	Resulting Action
OrderEvent	Start
SendOrderEvent	Start
ShipOrderEvent	Stop

Figure 8-61 OrderfulfillmentDuration stopwatch

Context: Stock Queries for Customer

In this section we will look at Stock Queries for customers see section “Create child monitoring contexts” on page 262 for more information.

1. Events

The only event for the Stock Queries for Customer context is the QueryEvent. It uses the QueryStock schema. See Figure 8-62 on page 268.

▼ Filter Condition
Define a condition based on the event attributes to identify whether to accept an event of this type.

QueryEvent/cicsEvent/cics:context-info/cics:eventname = 'QueryStock'

▼ Correlation Expression
Define an expression to identify the monitoring context instance or instances that receive the event at runtime.

../Customer_Number = fn:substring(QueryEvent/eventPayload/query:CustomerNumber,2) and
stockItem = fn:substring(QueryEvent/eventPayload/query:StockId,2)

If no instances are found: Create new instance
If one instance is found: Deliver to the instance
If multiple instances are found: Ignore

Figure 8-62 QueryEvent

Ensure that you correlate back up to the customer parent.

2. Set key

The key for this context is the Stock Item. Ensure that you strip off the leading + sign.

3. Counter

The counter for this context is Queries for Stock Item. It will be incremented for each new QueryEvent. This will allow us to track which stock items are queries the most frequently and so on.

Update Customer Context

We need to add a few more counters to the parent context.

► Customer Orders In Progress

This is incremented when an OrderEvent arrives and is decremented when the OrderShipped trigger (from the child context) fires.

► Customer Orders Shipped

This is incremented when the OrderShipped trigger (from the child context) fires.

This completes the monitor details model. We now move on to create some KPIs.

8.4.3 Create KPI and Dimensional Models

We now create some KPIs and dimensions that will allow us to perform some analysis on our monitored processes.

KPIs

A KPI context is a container for key performance indicators (KPIs). Unlike a monitoring context, a KPI context has no keys or metrics. However, you must create a KPI context as a container before you can create KPIs.

1. Switch to the KPI model. See Figure 8-63.

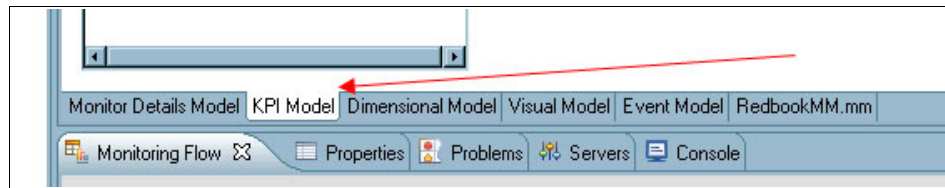


Figure 8-63 KPI model

2. Right-click in the left-hand pane and select **New KPI Context**.
3. Give this context a name (we chose Redbook).
We create two modeled KPIs.
4. Create Avg. Order Fulfillment Duration KPI.
 - a. Right-click and select **New KPI**.
 - b. Name the KPI (we chose Avg. Order fulfillment Duration).
 - c. Scroll down to KPI Definition.
 - d. Select the **Base this KPI on a metric and an aggregation function** radio button.
 - e. Select the **Orders for Customers** monitoring context.
 - f. Select the **Order fulfillment Duration** metric.
 - g. Select **Average** as the aggregation function.

Note: We have done something in our example that we possibly would not do in an actual implementation. That is, to use a stopwatch as the metric for a KPI. When possible, use duration metrics instead of stopwatches as the source of KPIs. KPIs based on stopwatches will not perform as well as KPIs based on duration metrics, because actively running stopwatch values are calculated in the database for each instance to be aggregated. Basing a KPI on a stopwatch also limits your options for tuning the database in the future. See Figure 8-64.

KPI Definition

Specify how the value of the KPI is set.

KPI Value

Choose how the KPI will get its value:

☒ Base this KPI on a metric and an aggregation function.

☐ Write an expression to calculate this KPI based on existing KPIs

KPI Details

Monitoring context:

* Orders for Customer

Browse...

Metric:

* Order Fulfillment Duration

Browse...

Aggregation function:

* Average

Use values from:

☒ All model versions☐ Only this version of the model

Time Filter

Select a time period over which the KPI should be calculated.

Metric:

Browse...

Time period:

☒ None☐ Repeating☐ Rolling☐ Fixed

Figure 8-64 Order for Customer KPI

- h. Set a target and some ranges for the KPI. We have set the target to five minutes in our example (which is only for the sake of expediency. A KPI target would normally be something along the lines of an SLA target, for example).
- i. Select the check-box to allow for historical tracking of KPIs, if required.

Note: This has the potential to be powerful. KPI values will be calculated hourly and kept in a KPI history table. In the KPI Manager widget in the Business Space, you will be able to create prediction models and use the collected historical data to predict future trends for the KPI.

5. Create Avg. Order Value KPI. See Figure 8-65.

▼ KPI Details

Edit the details of the KPI, which is a performance measurement used to track business objectives.

ID: Edit...

Name:

Description:

Type:

☒ Keep track of historical values for this KPI

▼ KPI Target and Ranges

Specify a target, which is an exact value for the KPI to achieve, or ranges against which to track the KPI, or both.

Target: Details...

Ranges:

Range name	Start value	End value	Color	
In Range	0 Milliseconds	< 5 Minutes		
Out of Range	5 Minutes	< 10 Minutes		
Alert Range	10 Minutes	< 15 Minutes		

Add Remove Sort

Figure 8-65 KPI target and ranges

a. Create another KPI of type decimal (Avg. Order Value).

KPI is based on a metric and an aggregation.

- Monitoring context = Orders for Customer
- Metric = Order Value
- Aggregation function = Average (Figure 8-66 on page 272)

KPI Definition
Specify how the value of the KPI is set.

KPI Value
Choose how the KPI will get its value:
☒ Base this KPI on a metric and an aggregation function.
☐ Write an expression to calculate this KPI based on existing KPIs

KPI Details
Monitoring context: * Orders for Customer Browse...
Metric: * Order Value Browse...
Aggregation function: * Average ▼
Use values from: ☒ All model versions ☐ Only this version of the model

Figure 8-66 Orders for Customer: Aggregation function and metric

- b. Set a target and ranges for the metric (we set a target of 10 for the purposes of this example).
- c. Set a currency from the drop-down menu, if you wish that to be shown in the dashboards. See Figure 8-67.

KPI Details
Edit the details of the KPI, which is a performance measurement used to track business objectives.

ID: * Avg_Order_Value Edit...
Name: Avg_Order_Value
Description:
Type: * Decimal
Currency: ▼ Decimal Precision: 0 ☐ Show as a percentage
☒ Keep AED - United Arab Emirates Dirham
AFN - Afghani
ALL - Albanian Lek
AMD - Armenian Dram

KPI Target and Ranges
Specify a target, which is an exact value for the KPI to achieve, or ranges against which to track the KPI, or both.

Target: 10 Details...
Ranges: * Actual value

Range name	Start value	End value	Color
Alert Range	0	< 8.5	
In Range	8.5	< 12	
Above Target	12	< 20	

Figure 8-67 Set currency and decimal precision

6. Save the new KPIs

Dimensions

When you create a monitoring context, an AlphaBlox® cube is automatically created and associated with the new monitoring context. The cube defines how information collected by monitoring context instances is stored.

Note: DB2 AlphaBlox cubes enable further data analysis and representation capabilities for the Business Monitor Dashboards. For more information regarding AlphaBlox, see the Info Center:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

Dimensions are data categories used to organize and select data for retrieval, monitoring, and analysis. Dimensions are composed of one or more hierarchical levels. For example, Location could be a dimension with levels of City, Region, and Country. You can define dimensions to be used in dimensional analysis in the dashboards.

Measures are calculations based on metrics. A measure points to a metric (such as order amount) and specifies an aggregation function (such as average or sum).

In the following steps, we create some measures and dimensions for the parent (customer monitoring context) and the children (order and stock queries) for use in analysis in the dashboards.

1. Move to the Dimensional Model. See Figure 8-68.

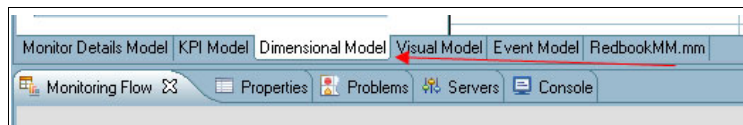


Figure 8-68 Dimensional model

2. Right-click the Customer MC and select **New Dimension**.
3. Select a name for the dimension (we chose Customer).
4. Click **OK**.
5. Right-click the newly created dimension and select **New Dimension Level**. See Figure 8-69 on page 274
6. The easiest way to create a dimension level is to click **Browse** and select the metric that you want to use.

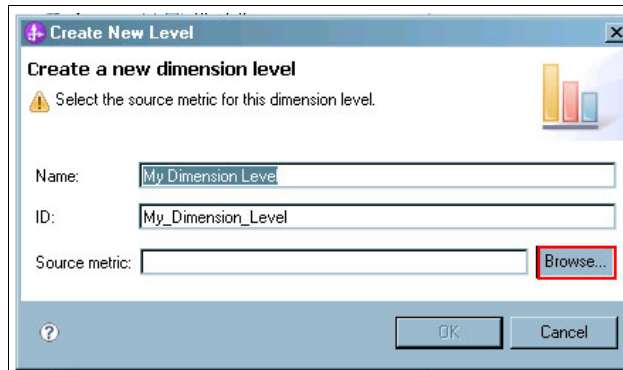


Figure 8-69 Create dimension level

7. Select the metric you need (in this case it is Customer Number).

The name of that metric will be used for the dimension level. See Figure 8-70.

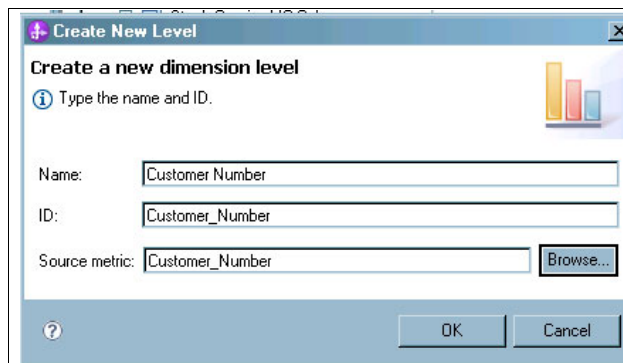


Figure 8-70 New dimension level

You may now see some errors start to appear. That is because, for a metric to be used in a dimension, it must be a mandatory metric.

8. Go back to the monitor details model and ensure that the metric is flagged as mandatory and had a default value. See Figure 8-71 on page 275.

Note: Set the default value to “rather than” or some other value.

☒ A value is required for this key

Default Value:

☐ This key can be used for sorting

Figure 8-71 Required metric

9. Go back to the dimensional model and create another dimension of Customer type which has the Premium as the metric for the dimension level 1.
10. Create some additional dimensions for the Orders and Stock Queries cubes as in Table 8-2.

Table 8-2 Orders and Stock Queries

Cube	Dimension	Metric
Orders	City	City
	Country	Country
	Customer Type	Premium
	Location	L1 Country L2 City
	Stock Item	Stock Item
Stock Query	Stock Item	Stock Item

11. Adjust the mandatory flags on all the source metrics so that there are no errors.

We create some measures for the dimensional analysis.

12. Right-click the Customer cube and select **New Measure**. See Figure 8-72 on page 276.

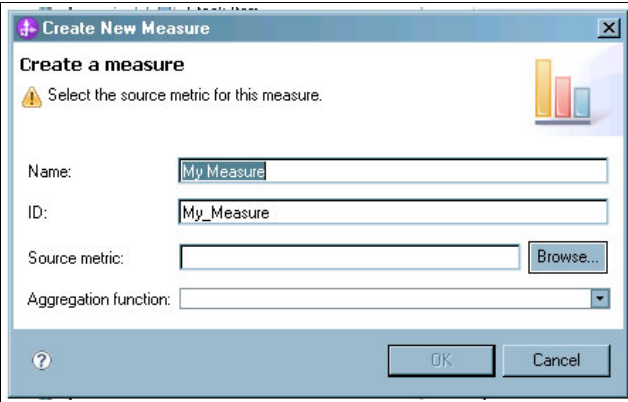


Figure 8-72 Create measure

- 13. Give the measure a name (we chose Stock Queries).
- 14. Select a source metric (we chose Customer Queries).
- 15. Select an aggregation function (we chose Sum).
- 16. Click **OK**.
- 17. Repeat this procedure to create additional measures for the Customer cube as shown in Figure 8-73.

Measures

Work with the measures for this cube. Measures are calculations based on a metric, key, counter, or stopwatch.

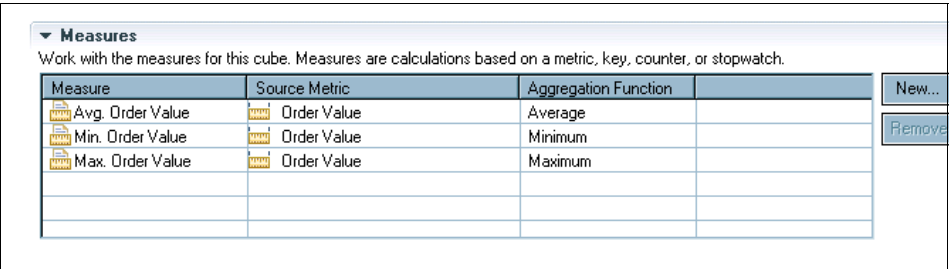
Measure	Source Metric	Aggregation Function	
Stock Queries	Customer Queries	Sum	
Orders Received	Customer Orders Received	Sum	
Orders Shipped	Customer Orders Shipped	Sum	

New...

Remove

Figure 8-73 Measures for Customer

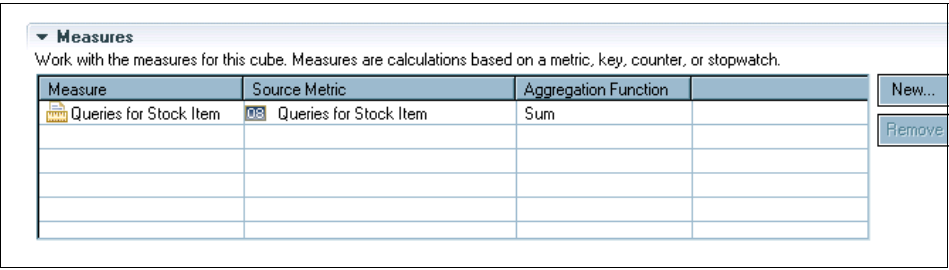
18.Create measures for the Orders cube as in Figure 8-74.



Measure	Source Metric	Aggregation Function	
Avg. Order Value	Order Value	Average	
Min. Order Value	Order Value	Minimum	
Max. Order Value	Order Value	Maximum	

Figure 8-74 Measures for Orders

19.Create the measures for the Queries cube as in Figure 8-75.



Measure	Source Metric	Aggregation Function	
Queries for Stock Item	Queries for Stock Item	Sum	

Figure 8-75 Measures for Queries

20.Save all of your work.

At this point it may be a good idea to run a quick validation of the monitor model.

21.Right-click the monitor model in the Project Explorer pane and select **Validate**. See Figure 8-76 on page 278.

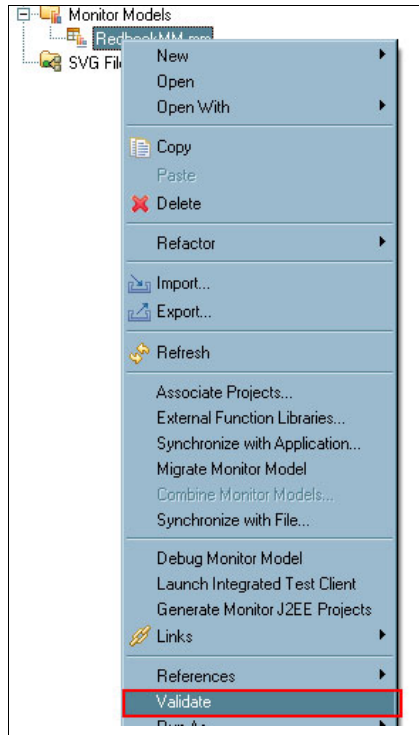


Figure 8-76 Validate monitor model

You should see a result shown in Figure 8-77.

If you have any errors, fix them before proceeding to the next section.

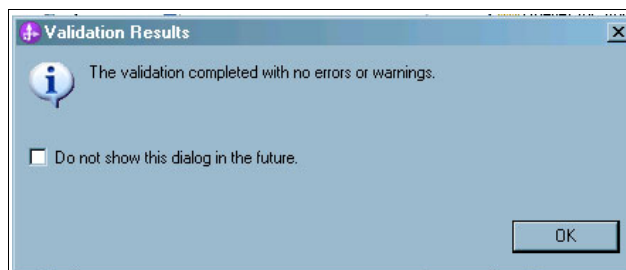


Figure 8-77 Successful validation

8.5 Generate and deploy monitor model

Now that we have created a monitor model, it is essential that we test it as part of our development. Such testing is performed on a running monitor model within the WebSphere Business Monitor test environment. In the test environment, results of the tests are displayed in Business Space where we can quickly verify that the metrics are being collected correctly.

Important: Normally, the monitor model would be deployed to a runtime environment after successful testing. As we mentioned earlier, we are using the test server that is installed with the Monitor Toolkit and will only deploy to our test environment. Should you wish to deploy your monitor model to a runtime environment, instructions on how to do this can be found in the InfoCenter at the following Web page:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

Before we can publish the monitor model to the test environment, we must first generate the J2EE™ projects from the model.

1. In the Project Explorer, right-click the model and select **Generate Monitor J2EE Projects**. See Figure 8-78.

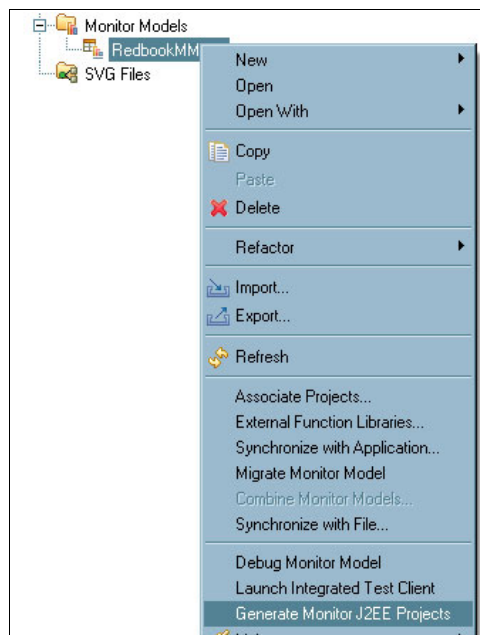


Figure 8-78 Generate Monitor J2EE Projects

2. Accept the default names.

Note: You can accept the default names for the projects, but be sure that the names are not too long. In Windows, there is a maximum path length of 256 characters, so any path longer than 128 characters at creation time will likely cause problems by the time it is published.

3. Click overwrite existing projects (if this is a regeneration of the monitor model).
4. Click **Finish**. See Figure 8-79.

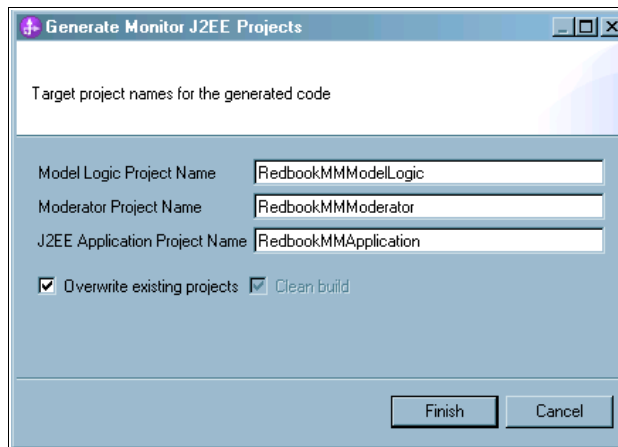


Figure 8-79 Generate projects, part 1 of 2

5. You will see a panel display, telling you that the generation has started.
6. If you want to continue working, you can click **Run In Background** and use the **Progress** button to check. See Figure 8-80.

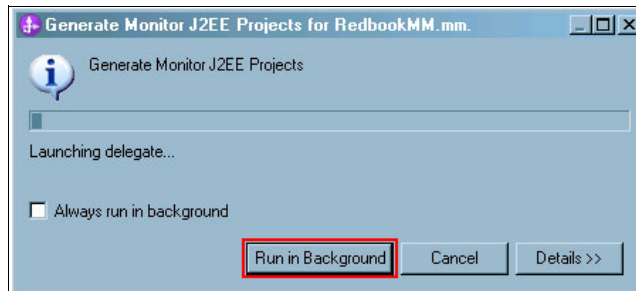


Figure 8-80 Generate projects, part 2 of 2

Note: Do not be too concerned if this seems to take a while the first time that you perform the generation.

Once the generation has completed, we can now deploy the monitor model application.

7. Right-click the Monitor Server in the servers pane.
8. If the server is not already started, click **Start**.
9. Once the server has started, right-click the server again.
10. Select **Add** and **Remove Projects**. See Figure 8-81.

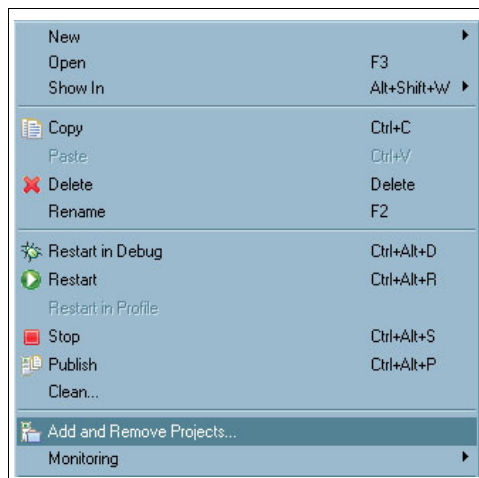


Figure 8-81 Add and Remove Projects

11. Select the RedbookMMAApplication.
12. Click **Add**. See Figure 8-82 on page 282.
13. Click **Finish**.

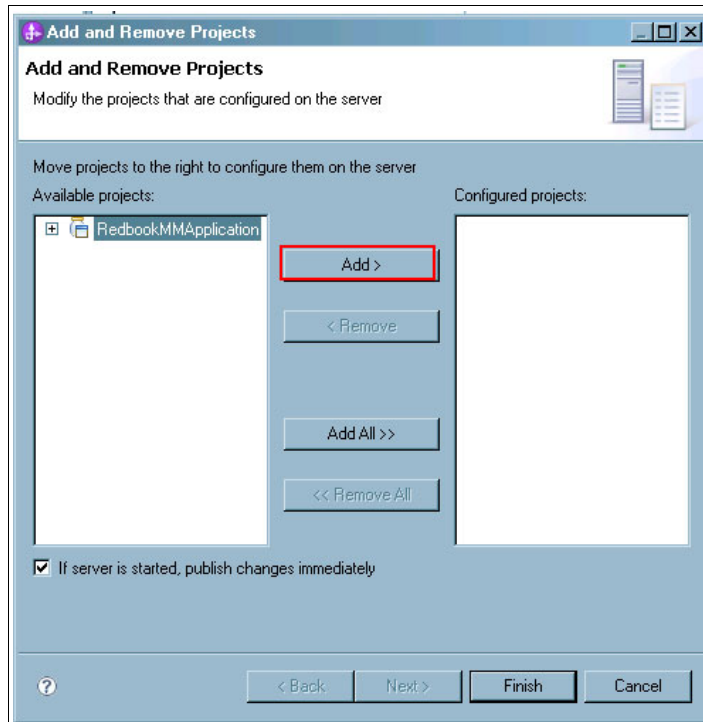


Figure 8-82 Add project to server

This will deploy the monitor model to the Monitor Server.

When you see the following message in the system log, the monitor model is ready to start processing events:

CWMRT3005I: The Monitor Model "RedbookMM 20090713143352" is starting consumption on this server or cluster member

Tip: The message you receive will not be identical to this. The name you will see here is the version of the monitor model being deployed. This is a combination of the monitor model ID and the monitor version time stamp from your monitor model.

8.6 Create Business Space dashboards

We now set up the dashboards in the Business Space to view our monitoring solution.

Business Space powered by WebSphere is a browser-based graphical user interface that lets business users interact with content from products in the WebSphere Business Process Management portfolio. The business spaces you create are collections of related Web content that provide you with insight into your business and the capability to react to changes in it.

Widgets are the pluggable user interface components that you use to define the functionality of your business spaces. For our example, we use the Business Monitoring widgets to create our dashboards.

Important: We have provided an export of the business space for the solution in the Additional Materials. The instructions we provide are to import and view this business space.

For information about how to create your own business spaces, see the WebSphere InfoCenter at the following Web page:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

1. Log on to the Business Space. See Figure 8-83 on page 284.

`http://<monitorhost:port/BusinessSpace/banner.jsp`

Note: In our example, we are logging on to an unsecured server, so a username is required, but no password.

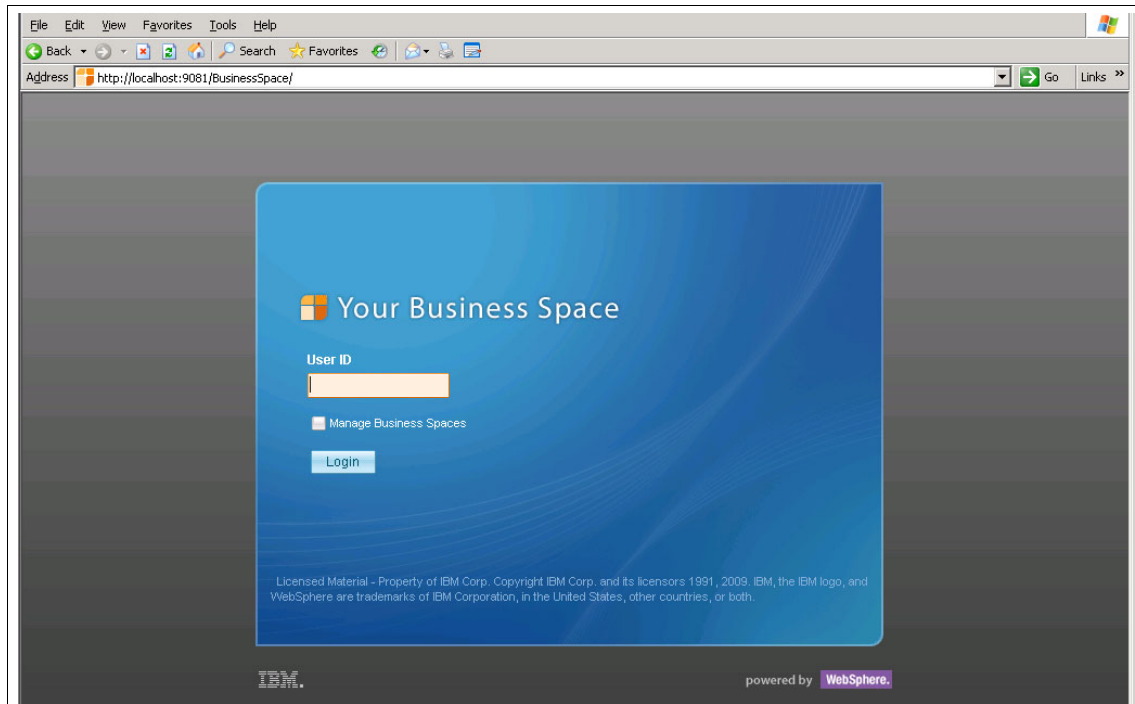


Figure 8-83 Business Space login panel

2. If you have not logged on previously, you will be directed to the Welcome page. Go to the right top corner and click **Manage Business Spaces**.
This will take you to the Business Space Manager, where you can create business spaces, edit settings for business spaces and import / export business spaces.
3. Select the **Import Business Space** icon as shown in Figure 8-84.



Figure 8-84 Import Business Space

4. Browse to the space.data (business space export file) from the Additional Materials. See Figure 8-85.

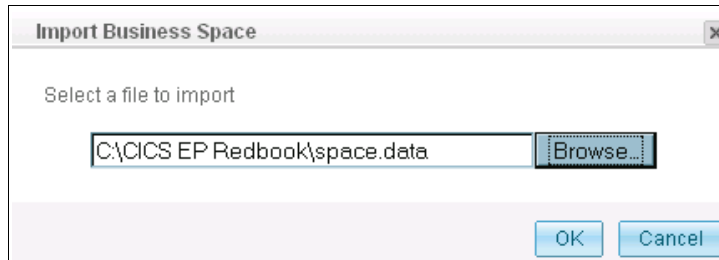


Figure 8-85 Browse to business space export file

5. Select **OK**.

This will import a business space named Redbook - CICS Events. See Figure 8-86.



Figure 8-86 New business space

6. Click the business space name to open the space.

A business space is made up of one or more pages, these pages are shown as tabs in the dashboard. Each of these pages contains one or more widgets. See Figure 8-87.

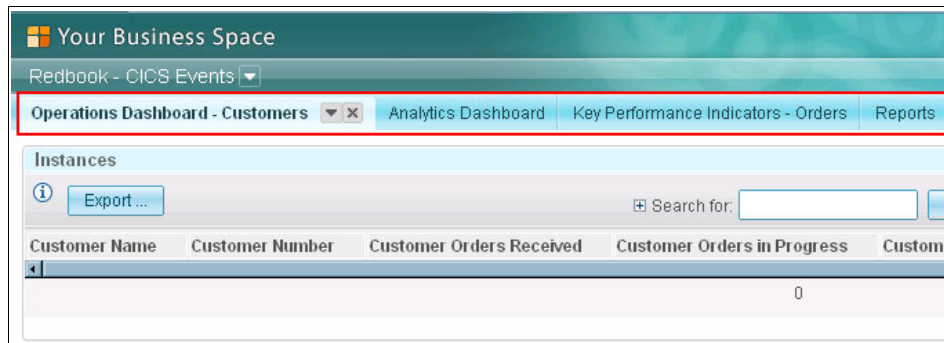


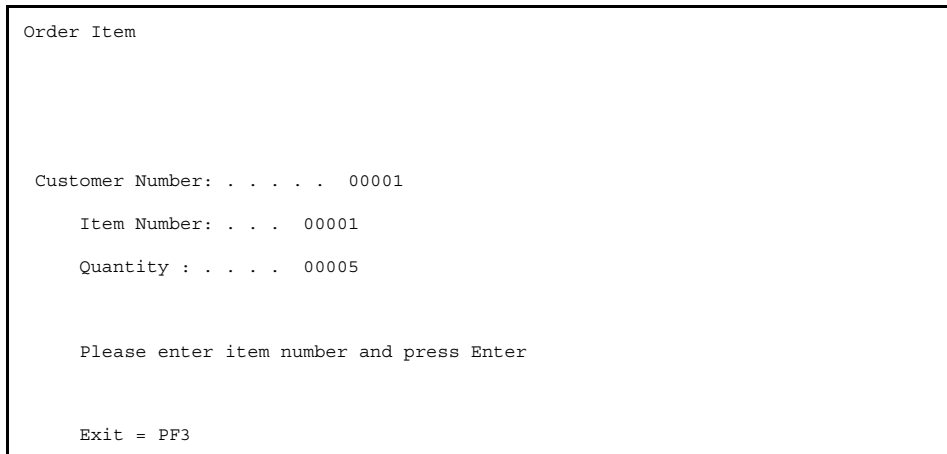
Figure 8-87 Pages in business space

At the moment we have no data in our business space, so we need to now go to CICS and generate some events.

8.7 Test and verify results

Log on to the sample shopping application (MENU).

1. Press Enter.
2. Enter customer number (we chose 00001).
3. Select PF2 (Order Item).
4. Enter an Item Number (we chose 00001) and Quantity (we chose 00005). See Figure 8-88.



```
Order Item

Customer Number: . . . . . 00001

Item Number: . . . 00001

Quantity : . . . . . 00005

Please enter item number and press Enter

Exit = PF3
```

Figure 8-88 Enter item number

5. Press Enter to place the order.
6. The order confirmation panel displays. See Figure 8-89.

Order Confirmation

Customer Number: 00001

Order Number: . . . 00034

Item Number: 00001

Quantity: 00005

Total Value of Order: . . . 9.95

PF3 = Menu Screen

PF12 = Exit

Figure 8-89 Order confirmation

Figure 8-90 on page 288 shows one of the actual CBEs that was received by the Monitor.

We can see the following information:

- ▶ The eventname is SendOrder (recall that this is the basis for the filtering and also forms part of the namespace of the payload)
- ▶ The usertag is V001 (this forms part of the namespace of the payload)
- ▶ The CustomerNumber is +00001, which is the basis for the correlation predicate.
- ▶ The OrderNumber is +00034, which is the basis for the correlation predicate of the child Orders for Customer MC.

```

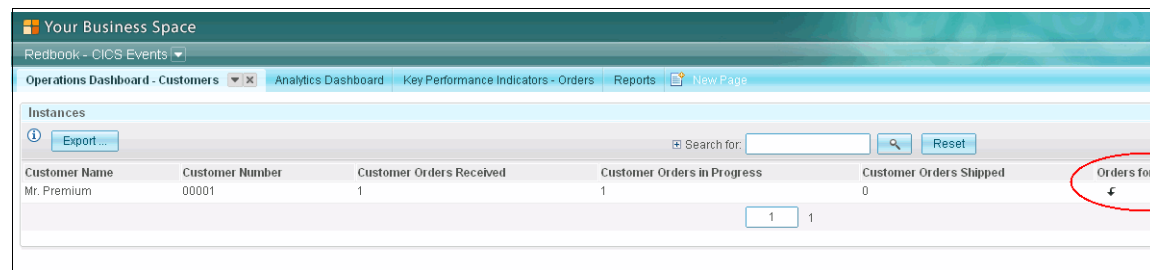
<CommonBaseEvent creationTime="2009-07-17T09:10:26.689+00:00" version="1.0.1">
  <sourceComponentId component="IBM CICS TS#4.1.0" componentIdType="ProductName"
  executionEnvironment="IBM z/OS" instanceId="USIBMSC.EPRED2" location="SC66" locationType="Hostname"
  subComponent="CICS EP" componentType="http://www.ibm.com/xmlns/prod/cics/eventprocessing"/>
  <situation categoryName="OtherSituation">
    <situationType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="OtherSituation" reasoningScope="EXTERNAL">
      <CICSApplicationEvent/>
    </situationType>
  </situation>
  <cics:event xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
    <cics:context-info>
      <cics:eventname>SendOrder</cics:eventname>
      <cics:usertag>V001</cics:usertag>
      <cics:networkapplid>USIBMSC.EPRED2</cics:networkapplid>
      <cics:timestamp>2009-07-17T09:10:26.689+00:00</cics:timestamp>
      <cics:bindingname>ShoppingMonitorBinding</cics:bindingname>
      <cics:capturespecname>CaptureSendOrder</cics:capturespecname>
      <cics:UOWid>1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F2F27F3721DC705B000100</cics:UOWid>
    </cics:context-info>
    <cics:payload-data>
      <data:payload xmlns:data="http://www.ibm.com/prod/cics/V001/SendOrder">
        <data:CustomerNumber>+00001</data:CustomerNumber>
        <data:OrderNumber>+00034</data:OrderNumber>
        <data:CustomerName>Mr. Premium </data:CustomerName>
        <data:City>Hursley </data:City>
        <data:Country>UK </data:Country>
        <data:Premium>Y</data:Premium>
        <data:TotalOrderValue>+000009.95</data:TotalOrderValue>
      </data:payload>
    </cics:payload-data>
  </cics:event>
</CommonBaseEvent>

```

Figure 8-90 SendOrder event

- Return to the Business Space. We now see that **Operational Dashboard - Customers** has changed to include the details for an order for customer 00001.

The Orders For Customer also now has an arrow which indicates that we now are able to drill-down to the child MC to see the orders for this customer. See Figure 8-91.



Customer Name	Customer Number	Customer Orders Received	Customer Orders in Progress	Customer Orders Shipped	Orders for
Mr. Premium	00001	1	1	0	1

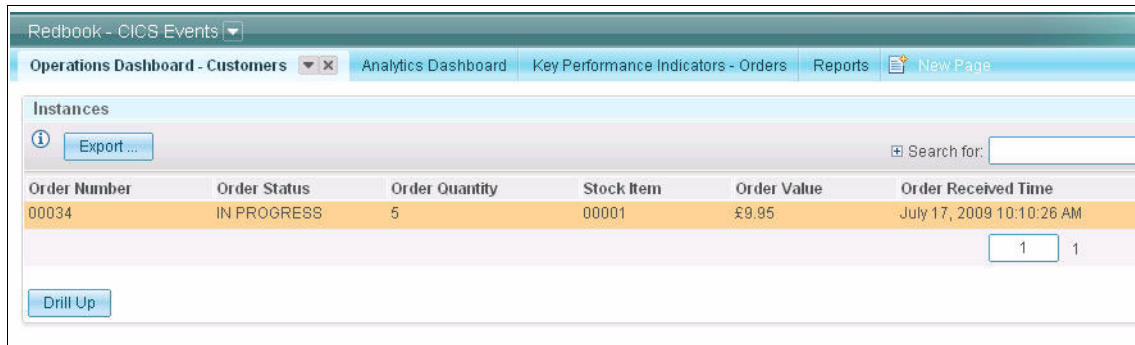
Figure 8-91 Customer information

- Click the arrow to drill-down to the order (if there were multiple orders we would see them all listed).

We see the order details (00034) that match in incoming events.

We also see the Order Received Time (the stopwatch tracking the duration of this order will also be running).

- Click Drill Up to return to the customer list. See Figure 8-92.



The screenshot shows a web application interface for 'Redbook - CICS Events'. At the top, there's a navigation bar with tabs: 'Operations Dashboard - Customers' (active), 'Analytics Dashboard', 'Key Performance Indicators - Orders', 'Reports', and 'New Page'. Below the navigation bar, there's a section titled 'Instances'. It contains an 'Export ...' button, a search bar labeled 'Search for:', and a table with the following data:

Order Number	Order Status	Order Quantity	Stock Item	Order Value	Order Received Time
00034	IN PROGRESS	5	00001	£9.95	July 17, 2009 10:10:26 AM

Below the table, there's a 'Drill Up' button and a small summary box showing '1' and '1'.

Figure 8-92 Order information

We also see some of the analytical data in the Analysis Dashboard tab.

Note: With only one order and one customer so far, this is not particularly interesting, but spend a few moments having a look at the possibilities.

Figure 8-93 on page 290 shows the analysis of the most popular items ordered (by country). From here you are able to see data (such as the number of orders (instances), average orders and so on). This gives some insight into how the buying patterns may be different across regional demographics.

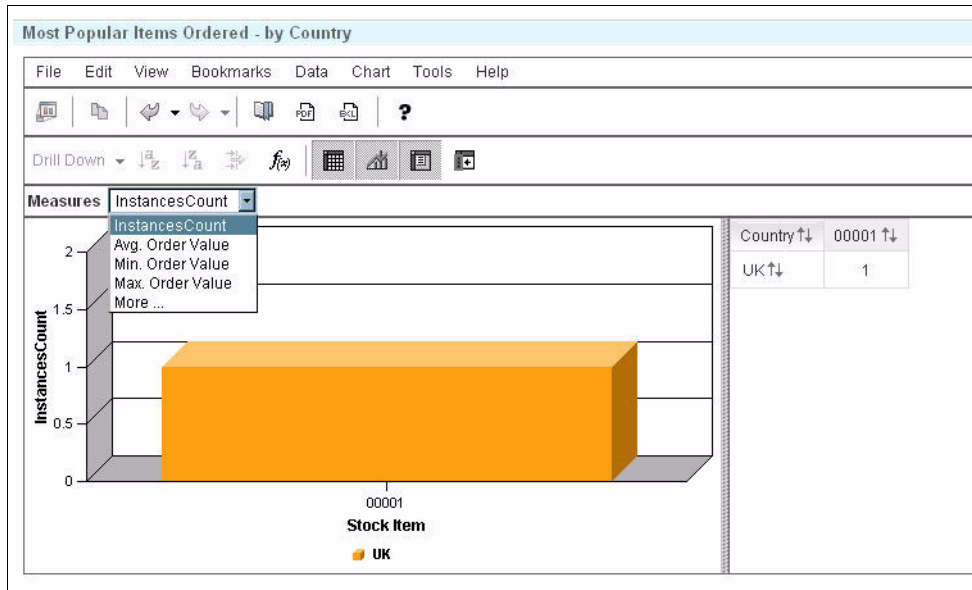


Figure 8-93 Most popular items

Figure 8-94 shows us the buying patterns, using the customer type as a demographic. We can slice this data to analyze the patterns of premium and non-premium customers across all countries or on a country by country basis.

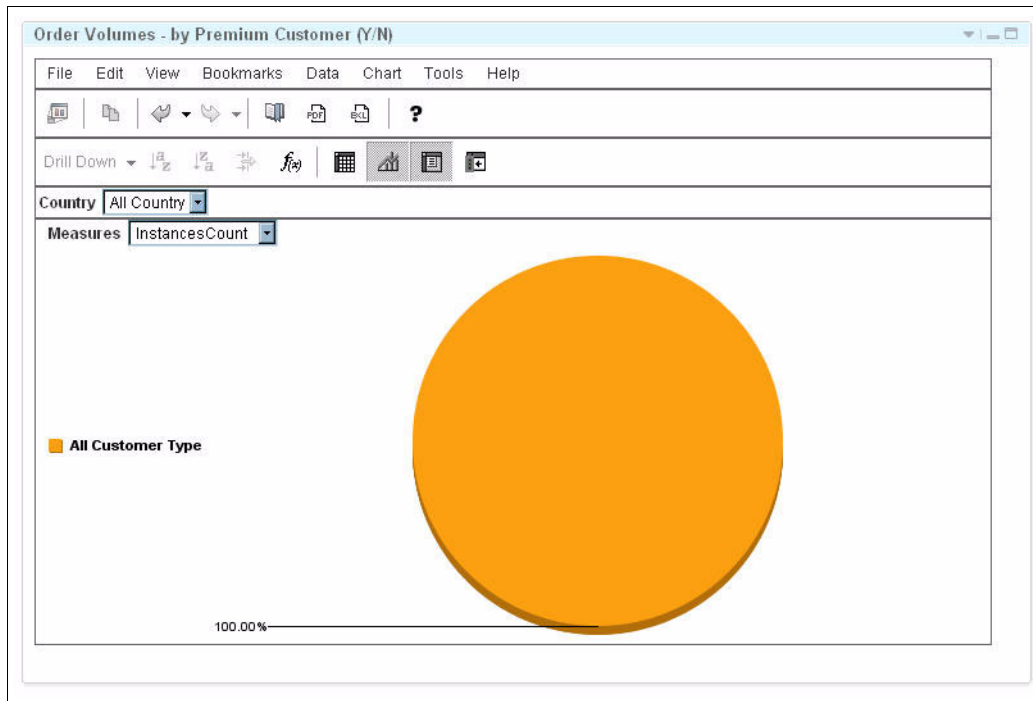


Figure 8-94 Customer type

10. Return to the Shopping application and select the option to ship the order.

Figure 8-95 on page 292 shows the ship event that arrived at the Monitor.

```

<CommonBaseEvent creationTime="2009-07-17T09:14:46.163+00:00" version="1.0.1">
  <sourceComponentId component="IBM CICS TS#4.1.0" componentIdType="ProductName"
  executionEnvironment="IBM z/OS" instanceId="USIBMSC.EPRED2" location="SC66" locationType="Hostname"
  subComponent="CICS EP" componentType="http://www.ibm.com/xmlns/prod/cics/eventprocessing"/>
  <situation categoryName="OtherSituation">
    <situationType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="OtherSituation" reasoningScope="EXTERNAL">
      <CICSApplicationEvent/>
    </situationType>
  </situation>
  <cics:event xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
    <cics:context-info>
      <cics:eventname>Ship</cics:eventname>
      <cics:usertag>V001</cics:usertag>
      <cics:networkapplid>USIBMSC.EPRED2</cics:networkapplid>
      <cics:timestamp>2009-07-17T09:10:46.163+00:00</cics:timestamp>
      <cics:bindingname>ShoppingMonitorBinding</cics:bindingname>
      <cics:capturespecname>CaptureShip</cics:capturespecname>
      <cics:UOWid>1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F2F27F3721DC705B000400</cics:UOWid>
    </cics:context-info>
    <cics:payload-data>
      <data:payload xmlns:data="http://www.ibm.com/prod/cics/V001/Ship">
        <data:OrderNumber>+00034</data:OrderNumber>
        <data:CustomerNumber>+00001</data:CustomerNumber>
        <data:StockId>+00001</data:StockId>
        <data:Quantity>+00005</data:Quantity>
      </data:payload>
    </cics:payload-data>
  </cics:event>
</CommonBaseEvent>

```

Figure 8-95 Ship event

11. Return to the **Operations Dashboard - Customers** and verify that the counter for orders in progress has been decremented and that the orders shipped counter has been incremented. See Figure 8-96.

Instances				
<div> <div> <div></div> <div>Export...</div> </div> </div>		<div> <div> <div></div> <div>Search for:</div> </div> </div>		
Customer Name	Customer Number	Customer Orders Received	Customer Orders in Progress	Customer Orders Shipped
Mr. Premium	00001	1	0	1
		<div> <div>1</div> <div>1</div> </div>		

Figure 8-96 Order shipped

12. Drill-down to the Order for Customer MC.
13. Verify that the Order Status has been updated to COMPLETED, the Order Shipped Time has been updated, and the stopwatch for the Order Fulfillment Duration has been stopped. See Figure 8-97.

Instances							
<div> Export Search for: <input type="text"/> Reset </div>							
Order Number	Order Status	Order Quantity	Stock Item	Order Value	Order Received Time	Order Shipped Time	Order Fulfillment Duration
00034	SHIPPED	5	00001	£9.95	July 17, 2009 10:10:26 AM	July 17, 2009 10:14:46 AM	4 m, 19.474 s
<div> 1 1 </div>							

Figure 8-97 Order shipped

14. Go to the Key Performance Indicators - Orders tab.
- We now see that the average order duration has changed, as has the average order value.
15. From Figure 8-98, we see that we are within our target for the end-to-end processing time for orders, but we are pretty much on target for our average value (the average is 9.95 and our target is 10).

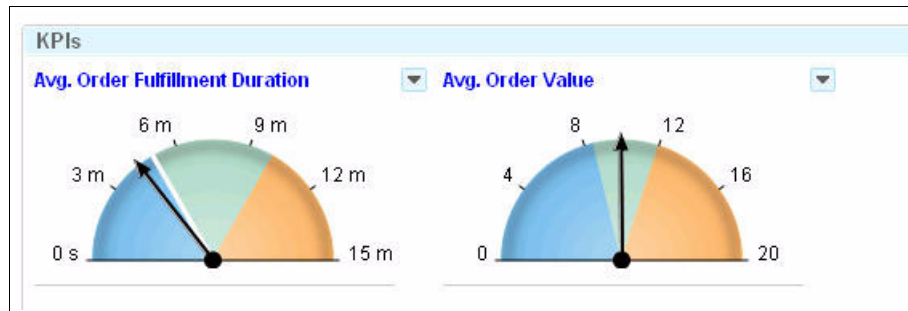


Figure 8-98 KPIs

16. Return to the shopping application and perform some stock query transactions.
- From Figure 8-99 on page 294, we can see that we have performed six queries (four of them for customer 00001, and two of them for customer 00005).
- We also now see that there is a black arrow in the Stock Queries for Customer column, indicating that we are able to drill-down to see which stock items each of the customers have queried.

Search for: <input type="text"/> <input type="button" value="Reset"/>			
Customer Number	Customer Queries	Orders for Customer	Stock Queries for Customer
00001	4	↓	↓
00005	2	↓	↓
1 - 2 2			

Figure 8-99 Stock queries

17.Drill-down to the stock queries for customer 00001.

We see that this customer has queried item 00001 twice and has also issued a query for item 0002 and item 0005. See Figure 8-100.

Stock Item	Queries for Stock Item
00001	2
00002	1
00005	1

Figure 8-100 Check queries

18. Go to the **Analytics Dashboard** in the Business Space.

We see that the “Most Frequently Queried Stock” panel is showing the new queries, indicating that stock item 00001 is our most frequently queried. See Figure 8-101.

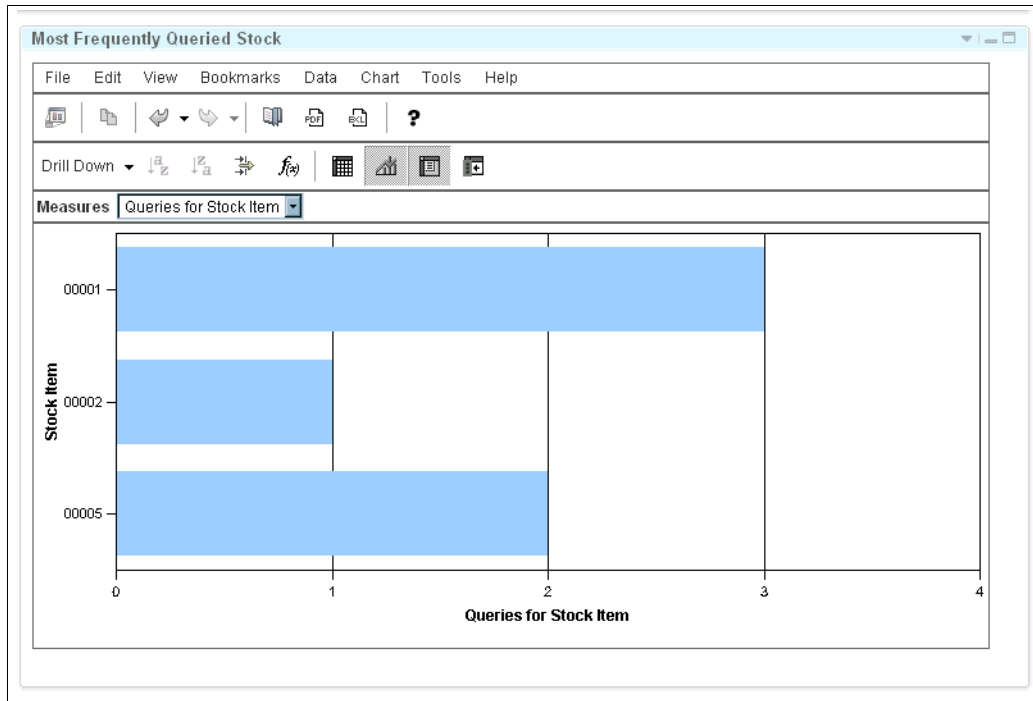


Figure 8-101 Stock queries

19. Return to the Key Performance Indicators - Orders tab and we see which of our customers have been placing orders (customer 0005 has made queries as we have seen earlier, but our analysis shows that he has not yet made any purchases.) See Figure 8-102 on page 296.

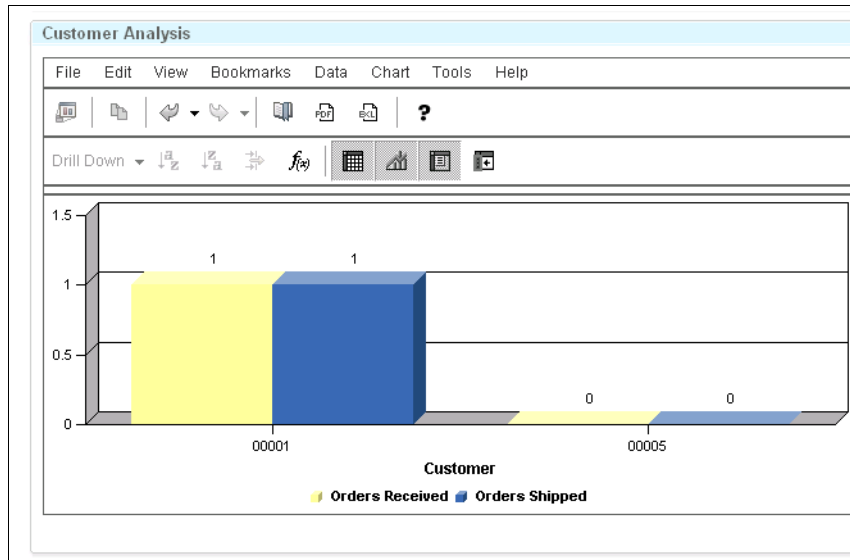


Figure 8-102 Customer analysis

8.8 Troubleshooting WebSphere Business Monitor

As we have seen in Chapter 6, “Troubleshooting” on page 149, there are things that can be checked to determine why an event may have not reached the Monitor server. In this section we discuss a few things that may go wrong once the event has reached the Monitor and how to troubleshoot them.

8.8.1 Events arrived from CICS but not delivered to monitor model

We have established (through previous troubleshooting methods) that the events have arrived at the Monitor server, but we do not see any evidence of them having been delivered to the monitor model.

1. Establish that your monitor model is in a started state and able to accept incoming events.
 - a. Log on to the Administrative Console of the Monitor server.
 - b. Select **Applications Monitor Models**. See Figure 8-103 on page 297.


```
<data:OrderNumber>+00034</data:OrderNumber>
<data:StockId>+00001</data:StockId>
<data:Quantity>+00005</data:Quantity>
</data:payload>
</cics:payload-data>
</cics:event>
</CommonBaseEvent>
```

The Monitor Toolkit also contains a debugging facility that you can use to validate whether the event arrives and whether the filter condition evaluates to true.

3. Go to the Monitor Toolkit.
4. Right-click the monitor model and select **Debug Monitor Model**. See Figure 8-104.

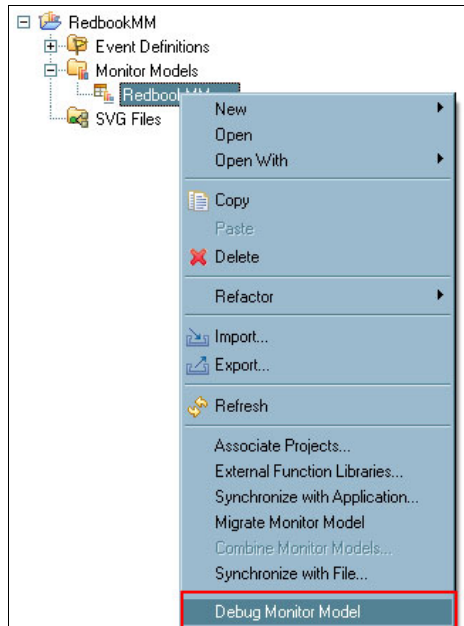


Figure 8-104 Start monitor debugger

This will open the debug perspective.

5. In the Detailed Steps pane, set breakpoints on all of the steps by right-clicking and selecting **Toggle Breakpoint**.

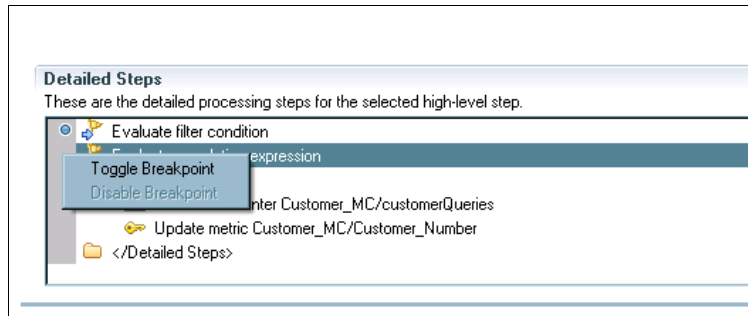


Figure 8-105 Turn on breakpoints

When the debug perspective opens, this will also open the Integrated Test Client. From here we send an event to the monitor model to test that our filter condition is correct.

6. Click the event part for cics:event in the Event Part Details pane.
7. Enter 0rder into the value pane. (This is the value that was present in the actual event, so we want to test it.)
8. We use this event for testing the correlation, so click the event:Payload event part and enter a value into the CustomerNumber (we chose +00009 as it is something we have not seen in our testing, but would be a valid value if received). See Figure 8-106 on page 300.

Integrated Test Client

Add Values to Events

Select an event definition and add test values. Click the Add to Test Script button to add the event instance to the test script.

Monitor model:

RedbookMM

Monitoring context:

Customer_MC

Event definition:

OrderEvent

Common base event data

Extension name:

Predefined data elements

Property data elements

Extended data elements

Event details

Event part details

Name	ID	Type	Path	
cicsEvent	cicsEvent	{http://www.ibm.co...}	cbe:CommonBaseEve...	
eventPayload	eventPayload	{http://www.ibm.co...}	cbe:CommonBaseEve...	

Name	Type	Value
cics:context-info/cics:eventname	string	Order

Add to Test Script

Add Edited Values

Cancel Edit

Figure 8-106 Test client event

300

Implementing Event Processing with CICS

9. Click **Add to Test Script**.

10. Click **Run Script** in the test script pane. See Figure 8-107.

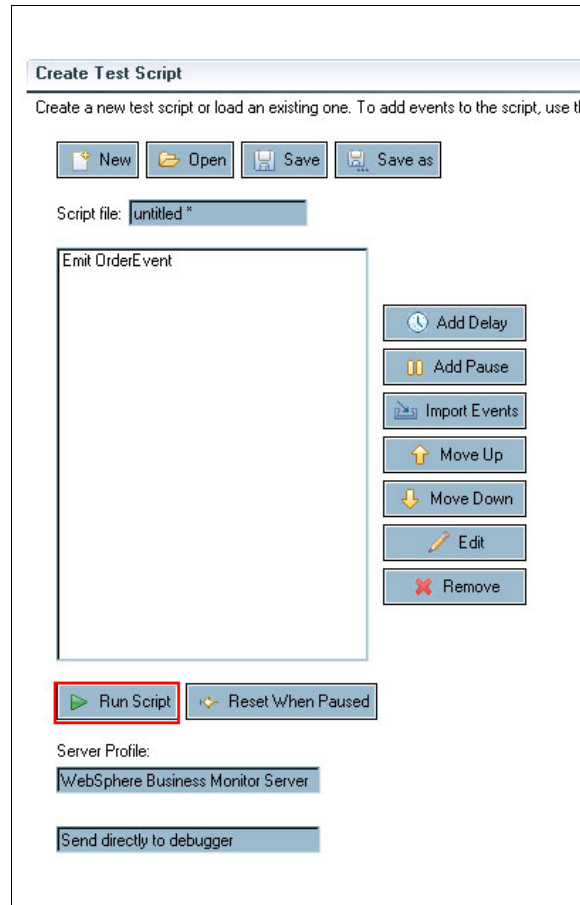


Figure 8-107 Send event

When the event has been emitted, the debugger will stop at the first breakpoint (which is to evaluate the filter condition).

11. Step into the processing by clicking the button as shown in Figure 8-108 on page 302.

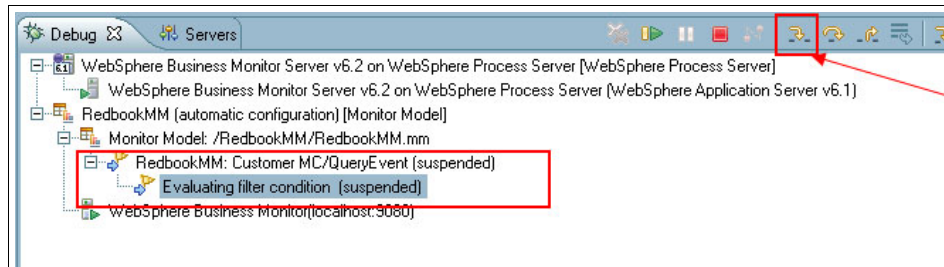


Figure 8-108 Evaluating filter condition

Figure 8-109 shows that the debugger has now stopped at the next breakpoint (which is to evaluate the correlation expression). This means that the filter condition has evaluated to true, the reason for this could be one of the following circumstances:

- ▶ If the filter condition evaluates to false, the rest of the steps are skipped, and processing continues with the next inbound event definition.
- ▶ If the filter condition evaluates to true, processing continues with the next step in the list of detailed steps.

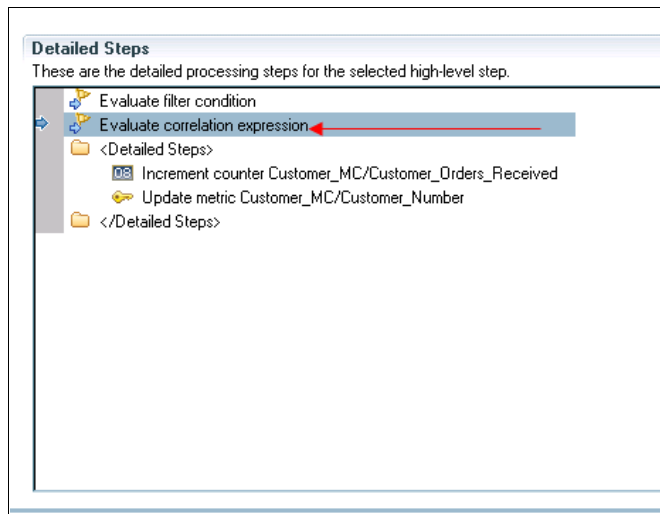


Figure 8-109 Filter evaluates to true

We see that the filter expression is being evaluated correctly.

Note: For more information about how to use the Monitor Model Debugger, see the InfoCenter at the following Web page:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp>

8.8.2 Events delivered to monitor model but not showing up in the dashboards

Another problem you may encounter is that the events, despite being delivered, are either not creating a monitoring context (MC) where you would expect or are not being added to a MC where you would expect. This may be a result of the correlation expression not evaluating the way you are expecting.

You can manually compare the correlation expression and key values against the actual event data. However, we continue with the debugger to show how to check that the correlation expression and key values are being processed correctly.

1. Step into the processing to evaluate the correlation expression.

In our example, this might result in the delivery of the event to one monitoring context instance. If no matching instance is found, because this is the first event we have seen for this customer, a new instance will be created and receive the event.

We see in Figure 8-110 that the correlation expression is processed successfully because of the following circumstances:

- If the event is not delivered to any instances, the rest of the detailed steps are skipped, and processing continues with the next inbound event definition. Processing of the event is suspended at the filter evaluation step of the next inbound event definition.
- If the event is delivered to at least one instance, continue with the next step.

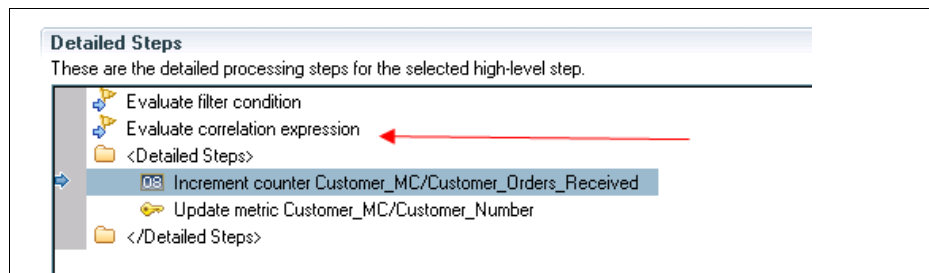


Figure 8-110 Successful correlation

2. Step into the breakpoints for the incrementing of the counter and the update of the key metric. See Figure 8-111.

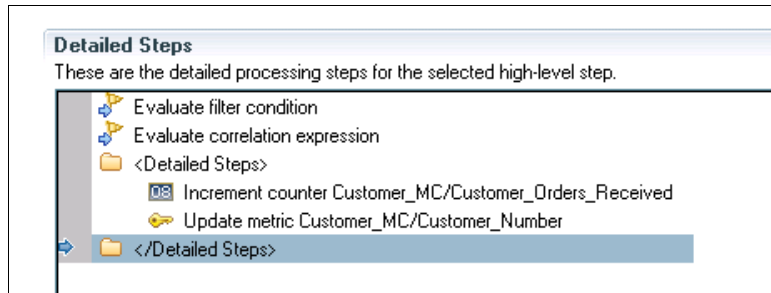


Figure 8-111 Stepping into processing

3. Click the **Instances view** (in the top right side pane).

We now see that the result of the event delivery was to create a new Customer MC instance and also to set the value of the key (which is the Customer Number) to 0009 (which is the customer number value we used in the test event, with the + sign stripped off as expected). See Figure 8-112.

Name	Value
Customer MC[a80498d6-bb9e-413b-b0db-49a16a49f864]	
customerQueries	0
Customer_Orders_Received	1
Customer_Orders_Shipped	0
Customer_Orders_in_Progress	0
Customer_Number	00009
Premium	

Figure 8-112 Instances

This shows that our correlation, event delivery and updating of key values is correct.

8.8.3 Metric values not being set or being incorrectly set

One other potential issue that you might encounter can arise from a problem with the schema declarations, such as in the namespaces. We encountered this problem when testing the monitor model against the events generated by CICS.

The expressions that are used to set the values of metrics are xpath. These xpath expressions traverse the event payload based on the event part definitions.

In one of our test runs, we discovered that we were receiving errors that indicated that the monitor model could not correctly find the event payload.

You will see in Figure 8-113 that there are certain values highlighted.

- ▶ eventname
- ▶ usertag
- ▶ payload

Note in the payload, the namespace URI for this event:

<http://www.ibm.com/prod/cics/V001/Order>

```
<CommonBaseEvent creationTime="2009-07-17T09:10:26.690+00:00" version="1.0.1">
  <sourceComponentId component="IBM CICS TS#4.1.0" componentIdType="ProductName"
  executionEnvironment="IBM z/OS" instanceId="USIBMSC.EPRED2" location="SC66" locationType="Hostname"
  subComponent="CICS EP" componentType="http://www.ibm.com/xmlns/prod/cics/eventprocessing"/>
  <situation categoryName="OtherSituation">
    <situationType xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:type="OtherSituation" reasoningScope="EXTERNAL">
      <CICSApplicationEvent/>
    </situationType>
  </situation>
  <cics:event xmlns:cics="http://www.ibm.com/xmlns/prod/cics/events/CBE">
    <cics:context-info>
      <cics:eventname>Order</cics:eventname>
      <cics:usertag>V001</cics:usertag>
      <cics:networkapplid>USIBMSC.EPRED2</cics:networkapplid>
      <cics:timestamp>2009-07-17T09:10:26.690+00:00</cics:timestamp>
      <cics:bindingname>ShoppingMonitorBinding</cics:bindingname>
      <cics:capturespecname>CaptureOrder</cics:capturespecname>
      <cics:UOWid>1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F2F27F3721DC705B000200</cics:UOWid>
    </cics:context-info>
    <cics:payload-data>
      <data:payload xmlns:data="http://www.ibm.com/prod/cics/V001/Order">
        <data:CustomerNumber>+00001</data:CustomerNumber>
        <data:OrderNumber>+00034</data:OrderNumber>
        <data:StockId>+00001</data:StockId>
        <data:Quantity>+00005</data:Quantity>
      </data:payload>
    </cics:payload-data>
  </cics:event>
</CommonBaseEvent>
```

Figure 8-113 Event from CICS

This namespace is not only contained in the event, but is also the namespace that is in the exported schemas that we use for the event part definitions in the monitor model. The namespace that is generated is as follows:

<http://www.ibm.com/prod/cics/<userTag>/<eventName>>

Figure 8-114 shows the original schema definition that we imported from an early version of ShoppingMonitorBinding.

```

<?xml version="1.0" encoding="UTF-8" ?>
<schema targetNamespace="http://www.ibm.com/prod/cics//Order"
xmlns:tns="http://www.ibm.com/prod/cics//Order" elementFormDefault="qualified"
attributeFormDefault="qualified" xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="payload">
    <complexType>
      <sequence>
        <element type="string" name="CustomerNumber"/>
        <element type="string" name="OrderNumber"/>
        <element type="string" name="StockId"/>
        <element type="string" name="Quantity"/>
      </sequence>
    </complexType>
  </element>
</schema>

```

Figure 8-114 Order.xsd

Notice that there is a subtle (but important) difference between the namespaces.

Having initially created an event binding without a userTag, we exported the event schemas and used them in the monitor model. Subsequently, someone changed the event binding to add a userTag, but omitted to re-export the schemas and update the event definitions in the monitor model.

This is how we come to have namespaces in Figure 8-114 that differ from the namespace seen in the actual event in Figure 8-113 on page 305.

When we imported the latest version of the schema (where the namespace definition correctly matched what was in the event), all of our metrics were correctly populated.



Custom EP adapters

In this chapter we discuss why you might need a custom EP adapter, describe how to write a custom EP adapter, and provide a sample custom EP adapter that emits simplified CICS events to WebSphere Business Monitor over HTTP.

9.1 Why write a custom EP adapter?

CICS TS V4.1 allows you to write your own, custom EP adapters. These are useful in the following situations:

- ▶ You want to format events in a way that is different from that provided by the CICS supplied EP adapters.
- ▶ You want to emit events through a different ‘transport’ from that provided by the CICS supplied EP adapters.
- ▶ You want to augment or enrich the event with information that your custom EP adapter can obtain.

9.2 How to write a custom EP adapter

CICS invokes an EP adapter to emit each event it captures, passing the EP adapter a channel containing one CICS event object. The EP adapter is a CICS command level capable program that takes a CICS event object as input, formats the data as required, and emits the formatted event to an event consumer.

The interface to EP adapters is a system programmer interface that may change from release to release.

EP adapters are not intended to be used as event consumers. That is, they are not intended to contain business logic. You should keep event-specific business logic in separate programs or transactions.

9.2.1 EP adapter samples

You can write a custom EP adapter in COBOL, Assembler, C, or PL/I. We have chosen to write the sample for this IBM Redbooks publication, EPCUSTOM, in COBOL, taking the sample custom EP adapter provided with CICS, DFH0EPAC, as a starting point.

DFH0EPAC emits events as single records to a TS queue. It differs from the CICS supplied TSQ EP adapter in that:

- ▶ It does not include context data in the event.
- ▶ It formats numeric data with leading blanks.
- ▶ It does not prefix positive numeric data items with a plus sign.

EPCUSTOM differs from DFH0EPAC in the following ways:

- ▶ It formats numeric data with leading zeroes.
- ▶ It formats events as XML data.
- ▶ It emits events using HTTP post.

WebSphere Business Monitor uses a monitor model to specify how information is to be extracted from an event at run time. It derives the model from schemas that define the structure of the event to be received (see 8.4, “Create monitor model” on page 235). When using the CICS WebSphere MQ EP adapter to create CBE events, export the schema describing the business information (the dynamic part of the payload of a CBE event) to WebSphere Business Monitor (see 5.12, “Creating events for WebSphere Business Monitor” on page 142).

Both the CICS supplied WebSphere MQ EP adapter and our sample, EPCUSTOM, format events for transmission to WebSphere Business Monitor. However, the CICS WebSphere MQ EP adapter emits events in CBE format to WebSphere MQ. However, the sample emits just the dynamic part of the payload of a CBE event over HTTP to the WebSphere Business Monitor REST listener, for the Event Emitter to build the CBE event.

For more information, see *WebSphere Business Monitor Event Emitter REST Interface*, at the following Web page:

<http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.bspace.620.help.api.rest.doc/apis/eventemitter/index.htm>

The payload created by EPCUSTOM conforms to the same schema as CBE events emitted by the CICS supplied WebSphere MQ EP adapter. This allows us to use these schemas to create monitoring models for EPCUSTOM events.

EPCUSTOM is not a production-ready EP adapter. Its limitations include the following restrictions:

- ▶ Its maximum event size is 32000 bytes but the CICS maximum is only restricted by storage availability.
- ▶ It does not escape the characters &,<,>,",' in XML data.
- ▶ Security requirements of WebSphere Business Monitor have not been considered.

9.2.2 The CICS event object

The CICS event object channel consists of the containers listed in Table 9-1.

Table 9-1 CICS event object containers

Container name	Description
DFHEP.ADAPTER	EP adapter configuration data as supplied on the Adapter tab of the event binding (user defined).
DFHEP.CONTEXT	Contextual data (described by DFHEPCXO).
DFHEP.DESRIPTOR	List of capture data items (described by DFHEPDEO).
DFHEP.DATA. <i>nnnnn</i>	A container for each captured data item where nnnnn is a 5 decimal digit sequence number that indicates the ordering of the captured data starting at '00001'

An EP adapter should first get the adapter, context, and descriptor containers using EXEC CICS GET CONTAINER.

Figure 9-1 on page 311 is an extract from EPCUSTOM, showing how to access these containers.


```

Working-storage section.
01 Program-working-storage.
    03 EPContextLength    pic s9(8) comp.
    03 EPDescriptorLength pic s9(8) comp.
    03 EPAdapterLength    pic s9(8) comp.
    03 Resp               pic s9(8) comp.
    03 Resp2              pic s9(8) comp.
    03 ResourceName       pic x(16).
*
Linkage section.
01 EPContext.
    copy dfhepcxo.
01 EPDescriptor.
    copy dfhepdeo.
01 EPAdapter      pic x(256).
*
Initial-processing section.
*   Obtain the DFHEP.CONTEXT container
    EXEC CICS GET CONTAINER('DFHEP.CONTEXT')
                SET(address of EPContext)
                FLENGTH(EPContextLength)

    END-EXEC.
*
*   Obtain the DFHEP.DESRIPTOR container
    EXEC CICS GET CONTAINER('DFHEP.DESRIPTOR')
                SET(address of EPDescriptor)
                FLENGTH(EPDescriptorLength)

    END-EXEC.
*
*   Obtain the DFHEP.ADAPTER container - if present
    EXEC CICS GET CONTAINER('DFHEP.ADAPTER')
                SET(address of EPAdapter)
                FLENGTH(EPAdapterLength)
                RESP(Resp) RESP2(Resp2)

    END-EXEC.
*
*   Determine the Resource name to use. Default to
*   EPCUSTOM if the adapter container is absent
    if Resp = dfhresp(normal) and EPAdapterLength > 0
        move EPAdapter(1:EPAdapterLength) to ResourceName
    else
        move 'EPCUSTOM' to ResourceName.

```

Figure 9-1 Accessing CICS event object containers

DFHEP.ADAPTER

The data in this container is a copy of that supplied in the “Data passed to the custom EP adapter” box on the Adapter tab in the event binding editor (see Figure 9-10 on page 326), converted to the local EBCDIC code page. The format of this data is up to you. We recommend you use it to specify formatting options and destination configuration data.

In our sample, we use it to specify the name of a URIMAP for the HTTP connection, defaulting it to EPCUSTOM if no adapter configuration is supplied.

DFHEP.CONTEXT

This container is described by copybook DFHEPCXO (see Figure 9-2 on page 313), which can be found in your CICS SDFHCOB library. Corresponding Assembler, C, and PL/I copybooks are supplied in SDFHMAC, SDFHC370, and SDFHPL1 respectively.

CICS captures certain items of contextual data at the time the event is captured and includes them in the event object in this container. They may be added to the emitted event if required, and used by the event consumer, for example, for event identification and correlation.

In our sample we add the Event Binding User Tag (epcx-ebusertag) and the Event Specification name (epcx-businessevent) to the XML namespace in the emitted event header, for example:

```
<data:payload xmlns:data="http://www.ibm.com/prod/cics/V001/SendOrder">
```

This is necessary to ensure events created by EPCUSTOM conform to the same schema as CBE events emitted by the CICS supplied EP adapter.

```

10 EPCX.
*   Structure identifier EPCX
   15 EPCX-STRUCID          PIC X(4).
      88 EPCX-STRUC-ID          VALUE 'EPCX'.
*   Structure version number
   15 EPCX-VERSION          PIC S9(8) COMP.
      88 EPCX-VERSION-1          VALUE 1.
      88 EPCX-CURRENT-VERSION    VALUE 1.
*   Schema version number
   15 EPCX-SCHEMA-VERSION    PIC S9(4) COMP.
      88 EPCX-SCHEMA-VERSION-1    VALUE 1.
      88 EPCX-CURRENT-SCHEMA-VERSION VALUE 1.
*   Schema release number
   15 EPCX-SCHEMA-RELEASE    PIC S9(4) COMP.
      88 EPCX-SCHEMA-RELEASE-0    VALUE 0.
      88 EPCX-CURRENT-SCHEMA-RELEASE VALUE 0.
*   Event Binding Name
   15 EPCX-EVENT-BINDING     PIC X(32).
*   Capture Secification name
   15 EPCX-CS-NAME           PIC X(32).
*   Event Binding user tag
   15 EPCX-EBUSERTAG         PIC X(8).
*   Business event name
   15 EPCX-BUSINESSEVENT     PIC X(32).
*   Transaction Id
   15 EPCX-TRANID            PIC X(4).
*   Current program name
   15 EPCX-PROGRAM           PIC X(8).
*   User Id
   15 EPCX-USERID            PIC X(8).
*   ABSTIME of event
   15 EPCX-ABSTIME           PIC S9(15) COMP-3.
*   Network Applid Qualifier
   15 EPCX-NETQUAL           PIC X(8).
*   Applid
   15 EPCX-APPLID            PIC X(8).
*   EIBRESP
   15 EPCX-RESP              PIC S9(8) COMP.
*   Network UOW Id
   15 EPCX-UOWID             PIC X(27).

```

Figure 9-2 *dfhepcxo*

DFHEP.DESRIPTOR

This container is described by copybook DFHEPDEO which can be found in your CICS SDFHCOB library.

The descriptor has a prefix which includes field epde-itemcount, the number of data items in the event, followed by an epde-item entry per data item. See Figure 9-3 and Figure 9-4 on page 315.

```
10 EPDE.  
*   Fixed length prefix  
15 EPDE-PREFIX.  
*   Structure identifier EPDE  
20 EPDE-STRUCID PIC X(4).  
   88 EPDE-STRUC-ID      VALUE 'EPDE'.  
*   Version of this structure  
20 EPDE-VERSION PIC S9(8) COMP.  
   88 EPDE-VERSION-1     VALUE 1.  
   88 EPDE-CURRENT-VERSION VALUE 1.  
*   Length of a data item  
20 EPDE-ITEMLENGTH PIC S9(4) COMP.  
*   Number of data items  
20 EPDE-ITEMCOUNT PIC S9(4) COMP.
```

Figure 9-3 dfhepdeo prefix

```

*   Data descriptor array item
15 EPDE-ITEM OCCURS 0 TO 999 DEPENDING ON EPDE-ITEMCOUNT.
*   Data item name
20 EPDE-DATANAME PIC X(32).
*   Data type code
20 EPDE-DATATYPE PIC X(8).
88 EPDE-PACKED      VALUE 'PACKED'  '.
88 EPDE-ZONED      VALUE 'ZONED'   '.
88 EPDE-HEX        VALUE 'HEX'     '.
88 EPDE-UHWORD     VALUE 'UHWORD'  '.
88 EPDE-UFWORD     VALUE 'UFWORD'  '.
88 EPDE-SHWORD     VALUE 'SHWORD'  '.
88 EPDE-SFWORD     VALUE 'SFWORD'  '.
88 EPDE-CHAR       VALUE 'CHAR'    '.
*   Data precision
20 EPDE-DATAPRECISION PIC S9(8) COMP.
*   Formatting data type
20 EPDE-FORMATTYPE PIC X(16).
88 EPDE-TEXT       VALUE 'text'     '.
88 EPDE-NUMERIC    VALUE 'numeric'  '.
*   Formatting length
20 EPDE-FORMATLEN PIC S9(8) COMP.
*   Formatting precision
20 EPDE-FORMATPRECISION PIC S9(8) COMP.

```

Figure 9-4 *dfhepdeo item array*

The EP adapter iterates through the epde-item entries, getting and formatting each captured data item.

So, for epde-item(1) we need to get the data from container DFHEP.DATA.00001, for epde-item(2) we get data from container DFHEP.DATA.00002, and so on.

Figure 9-5 is an extract from EPCUSTOM showing how to access the data containers.

```
Working-storage section.
01 Program-working-storage.
    03 EPDataLength      pic s9(8) comp.
    03 Resp              pic s9(8) comp.
    03 Resp2             pic s9(8) comp.
    03 ItemNum           pic 9(5).
    03 ContainerName     pic x(16).
*
Linkage section.
01 EPDescriptor.
    copy dfhepdeo.
01 EPData              pic x(32000).
*
Main-program section.
    perform Process-data-item
        varying ItemNum from 1 by 1
        until ItemNum > epde-itemcount.
*
Process-data-item section.
*   Build the data container name: DFHEP.DATA.nnnnn
    string 'DFHEP.DATA.' delimited by size
        ItemNum delimited by size
        into ContainerName
    end-string.
*
*   Obtain the DFHEP.DATA.nnnnn container - if present
    EXEC CICS GET CONTAINER(ContainerName)
        SET(address of EPData)
        FLENGTH(EPDataLength)
        RESP(Resp) RESP2(Resp2)

    END-EXEC.
```

Figure 9-5 Accessing CICS event object data containers

Notes:

- ▶ It is possible for a specified data item to be missing from the event because CICS was unable to capture it for some reason. In this case, the epde-item is present but the corresponding container will not be found.

In our sample, we replace such data items with asterisks but you should check that this behavior is expected in your environment if it occurs. It is likely that the event binding is incorrectly specified. For example, CICS cannot capture data if you specify a start offset that is greater than the length of the source data area. This may occur if a capture specification intended for one circumstance is inadvertently applied to a different one because the filters are not sufficiently well-specified.

- ▶ The length of the captured data can be deduced from the datatype (in the case of halfword and fullword items) or can be obtained from the length of the data container.
- ▶ The length of a data item is restricted only by the length of the data area from which it is captured but we define EPData as 32000 bytes, because that is the maximum event length we support in the sample.

DFHEP.DATA.nnnnn

There is one data container per captured data item.

Numeric data is captured in binary, as is, with no conversion.

CICS event processing assumes captured character data is in the same code page as that specified by the LOCALCCSID SIT parameter, unless you specify an alternate code page for the Information Source in the capture specification (see Figure 9-6 on page 319).

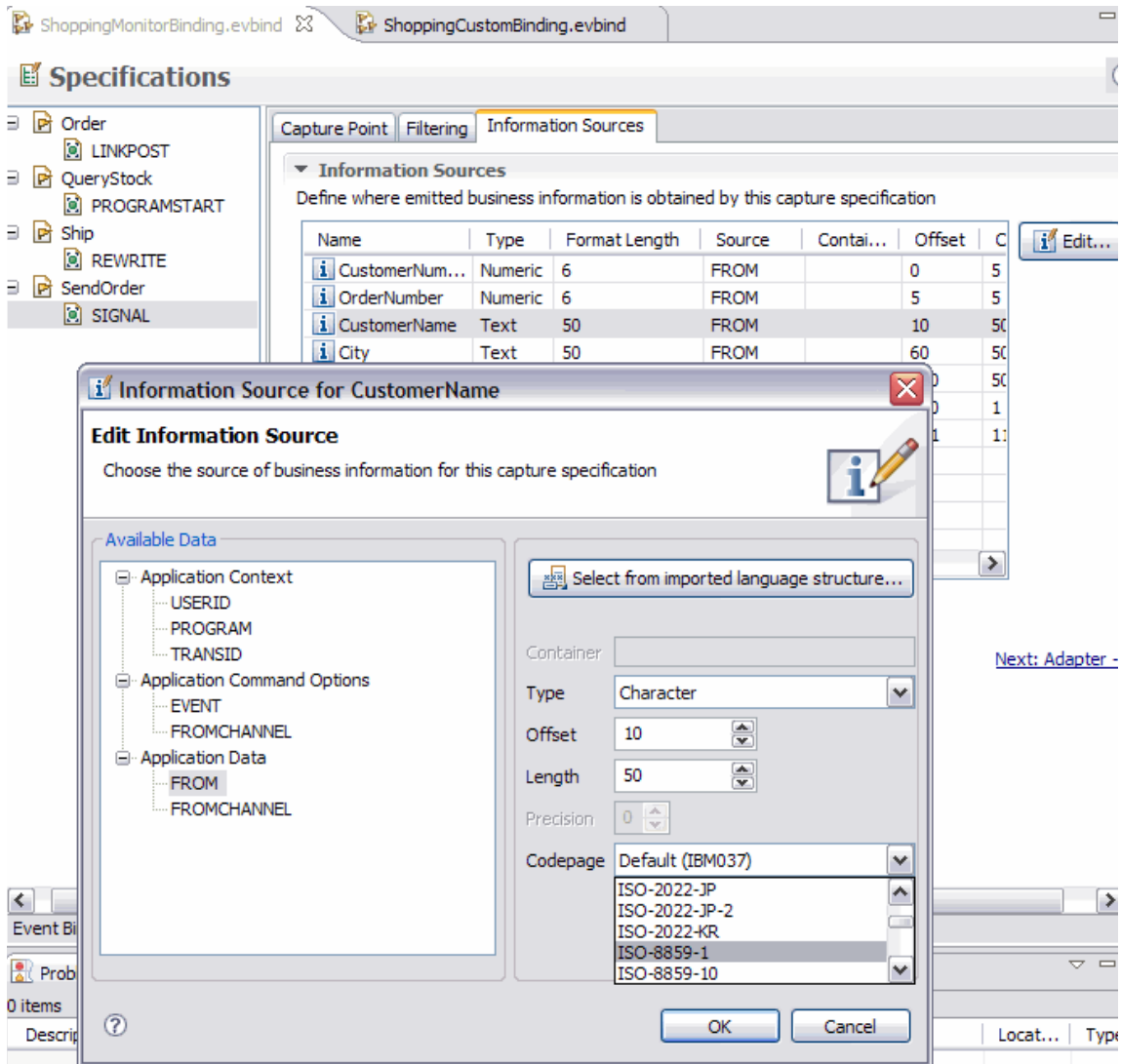


Figure 9-6 Overriding default codepage

CICS associates the codepage, if set, with the captured character data. When we retrieve character data in the EP adapter using the default options of the **EXEC CICS GET CONTAINER** command, CICS converts it to the LOCALCCSID.

9.2.3 Event formatting

Each epde-item in the DFHEP.DESRIPTOR container gives information about the captured data and how to format it. Fields epde-datatype and epde-dataprecision describe the captured data. Fields epde-formattype, epde-formatlength, and epde-formatprecision describe the formatting requirements.

Figure 9-4 on page 315 shows the supported data capture types. You can format the event data as you choose in a custom EP adapter. Our sample, EPCUSTOM, includes routines to format each data type, copied from the CICS sample EP adapter, DFH0EPAC:

1. Firstly, it converts the binary data to character according to its epde-datatype, using the Convert-data routine.
2. Then we are in a position to calculate the length of the output data item using the calculate-length routine. Format length may be specified explicitly, or set to zero, meaning that the EP adapter should determine an appropriate output length.

When format length is zero, the EP adapter determines the length of the event based on the length, and possibly content, of the captured data. This calculation is dependent upon the formatting algorithm, for example:

- Are numeric fields fixed length or just sufficient for the number of digits in the actual data captured for each event?
 - Is a sign added to positive numeric data?
 - Does a leading zero need to be added?
 - How does it handle missing data items?
 - Does character data need to be escaped for an XML format event?
3. EPCUSTOM formats each data item as a named data item in the payload of a CBE event (see Figure 9-7 on page 320). The data names (for example CustomerNumber) are the Emitted Business Information field names (see Figure 9-6). The EP adapter gets the names from the DFHEP.DESRIPTOR container field epde-dataname.

```

<data:payload xmlns:data="http://www.ibm.com/prod/cics/V001/SendOrder">
  <data:CustomerNumber>000005</data:CustomerNumber>
  <data:OrderNumber>000020</data:OrderNumber>
  <data:CustomerName>NAME5                                </data:CustomerName>
  <data:City>ADDR3                                         </data:City>
  <data:Country>ADDR4                                     </data:Country>
  <data:Premium>Y</data:Premium>
  <data:TotalOrderValue>0000000069.90</data:TotalOrderValue>
</data:payload>

```

Figure 9-7 An example event formatted by EPCUSTOM

This is necessary to ensure events created by EPCUSTOM conform to the same schema as CBE events emitted by the CICS supplied EP adapter.

EPCUSTOM adds the length of the XML tags to the length of the associated output data item to derive the full length required for the output field.

4. Finally, EPCUSTOM formats the data item into the buffer using the Format-data routine. Whatever the capture data type, we format the data according to its epde-formattype, as follows:
 - When it has the value text, we left-justify and pad with blanks.
 - When it has the value numeric, we right-justify and pad with zeroes (unlike the equivalent routine in DFHOEPAC, which pads numeric data with blanks).

The sample uses a fixed buffer and restricts the maximum size of events it emits to 32000 bytes. If you need to support large events you will need to decide whether to have a fixed buffer and allow for occasional overflow, or to do two passes of the descriptor, and possibly the data containers, to determine the event size to getmain a buffer of the appropriate size. If the format length is always specified, it should be a simple matter of iterating through the descriptor to calculate the buffer size required.

EPCUSTOM creates event data that can be included in a CBE event by the WebSphere Business Monitor REST listener. It builds a header, as described in “DFHEP.CONTEXT” on page 312, formats each captured data item as described above, and adds the XML end tag.

9.2.4 Event emission

You can emit, or not, the event to as many destinations as you choose in a custom EP adapter. See Figure 9-8.

1. EPCUSTOM writes the formatted event to a temporary storage queue whose name is the same as that of the URIMAP. We ignore errors as the WRITEQ TS is only for debugging purposes and, for example, the TS queue will eventually fill up if not purged.
2. EPCUSTOM uses the CICS WEB API to emit the event:
 - a. It opens a session to the WebSphere Business Monitor REST listener whose host and port are specified in the URIMAP. The EP adapter gets the name of the URIMAP from the DFHEP.ADAPTER container.
 - b. It does a WEB SEND to emit the event using an HTTP POST.
 - c. It closes the web session.

Where the default behaviour has a suitable effect, we omit error checking in EPCUSTOM to keep the sample short and easy to understand.

```
*      Open a connection to the WBM REST listener
EXEC CICS WEB OPEN URIMAP(ResourceName)
                        SESSTOKEN(SessionToken)
END-EXEC.

*
*      Post the event to the WBM REST listener
EXEC CICS WEB SEND POST
                        FROM(EventData)
                        FROMLENGTH(EventDataFlength)
                        SESSTOKEN(SessionToken)
                        MEDIATYPE('text/xml')
                        CHARACTERSET('ISO-8859-1')
END-EXEC.

*
*      Close the Web session
EXEC CICS WEB CLOSE SESSTOKEN(SessionToken)
END-EXEC.
```

Figure 9-8 Emitting the event

9.3 Running the sample EP adapter

In this section we show how to setup the CICS system to use the sample EP custom EP adapter, EPCUSTOM, instead of the CICS supplied WebSphere MQ EP adapter, to emit the same events to WebSphere Business Monitor (see 5.12, “Creating events for WebSphere Business Monitor” on page 142). We create the new artifacts shown in Table 9-2.

Table 9-2 Artifacts for running the sample EP adapter

Name	Description
EPCUSTOM	EP adapter load module
ShoppingCustomBundle	Explorer bundle project for event binding
ShoppingCustomBinding	Explorer event binding resource
SHOPCUST	BUNDLE definition for exported ShoppingCustomBundle
EPCUSTOM	PROGRAM definition for the EP adapter
EPCA	TRANSACTION definition for the EP adapter
EPURIMAP	URIMAP definition for the WebSphere Business Monitor REST listener

9.3.1 EP adapter program EPCUSTOM

We supply the source of EPCUSTOM in the additional material available with this IBM Redbooks publication. You can translate, compile, and link it into a DFHRPL library using your standard CICS COBOL compilation procedures.

9.3.2 Event binding ShoppingCustomBinding

Because we want to send the same events to WebSphere Business Monitor as we did in 5.12, “Creating events for WebSphere Business Monitor” on page 142, we can either modify the ShoppingMonitorBinding event binding created previously, or copy and modify it. We copy and modify it so that both event bindings can be made available as additional material with this Redbooks publication.

Before doing so, ensure that schemas for the events have been exported to WebSphere Business Monitor. As the EPCUSTOM event payload conforms to the same XML schema as the payload of an equivalent CICS CBE event transmitted over WebSphere MQ, you can use the same event schema to generate a monitor model.

If you have not already exported the schemas, perform the following steps:

1. In the Project Explorer window of the Resource perspective in the CICS Explorer, expand the ShoppingMonitorBundle project and double-click ShoppingMonitorBinding to open it. See Figure 9-9 on page 324.
2. Click the Adapter tab at the bottom of the window.
3. Click **Export Event Specifications**.

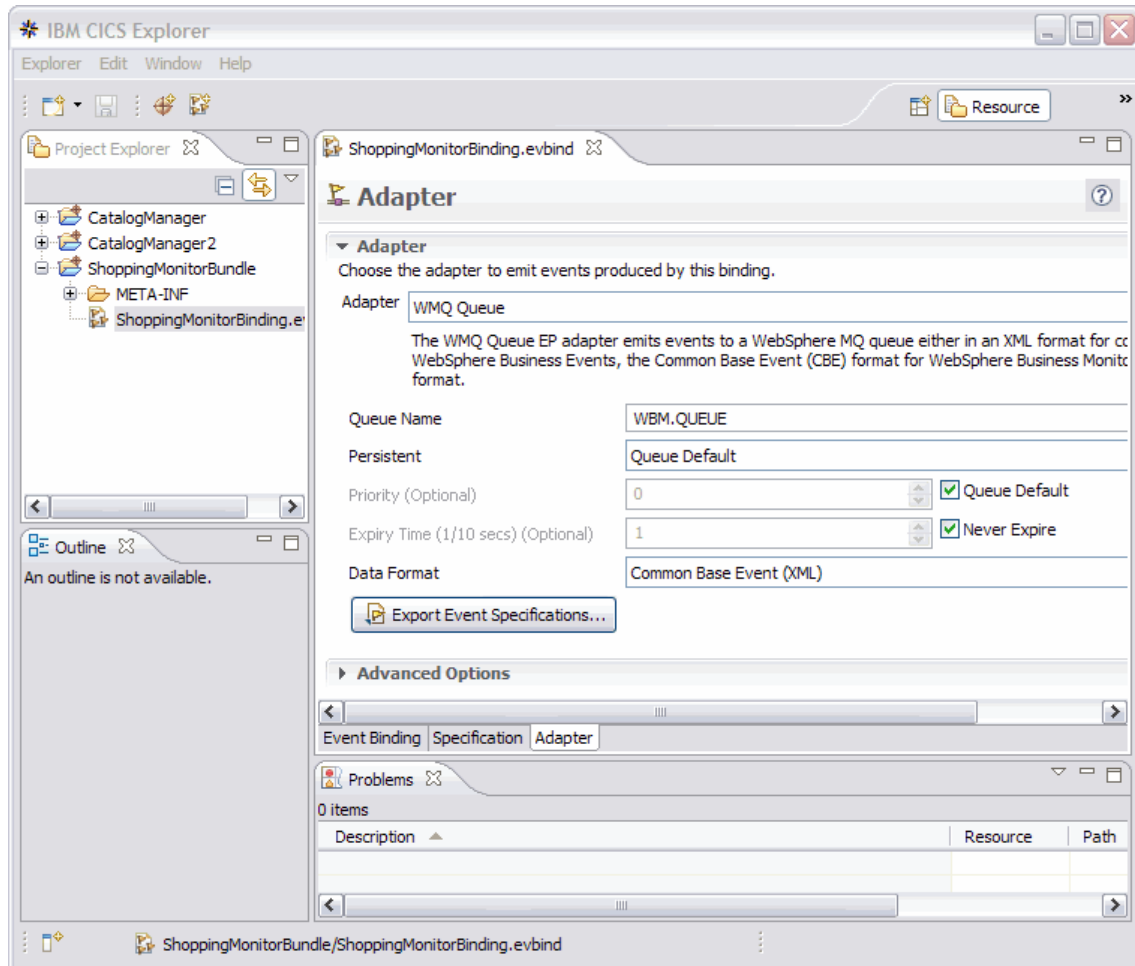


Figure 9-9 Export schemas for monitoring model

Then create a new event binding which will use the custom EP adapter.

4. Copy the ShoppingMonitorBundle and paste it as a new bundle project, ShoppingCustomBundle.
5. Expand ShoppingCustomBundle to view the project contents and rename ShoppingMonitorBinding as ShoppingCustomBinding.

6. Double-click **ShoppingCustomBinding.evbind** to edit it and select the Adapter tag. See Figure 9-10 on page 326.

Select the **Custom (User Written)** option from the Adapter drop-down menu. Supply a transaction ID for the EP adapter to run under and a URIMAP, which will be used to access the WebSphere Business Monitor REST listener. We use the following values:

- EPCA as the Transaction ID.

This tells CICS which EP adapter program to run.

- EPURIMAP as Data passed to the custom EP adapter.

This data will appear in the DFHEP.ADAPTER container. The sample EP adapter expects the name of the URIMAP it can use to emit the event.

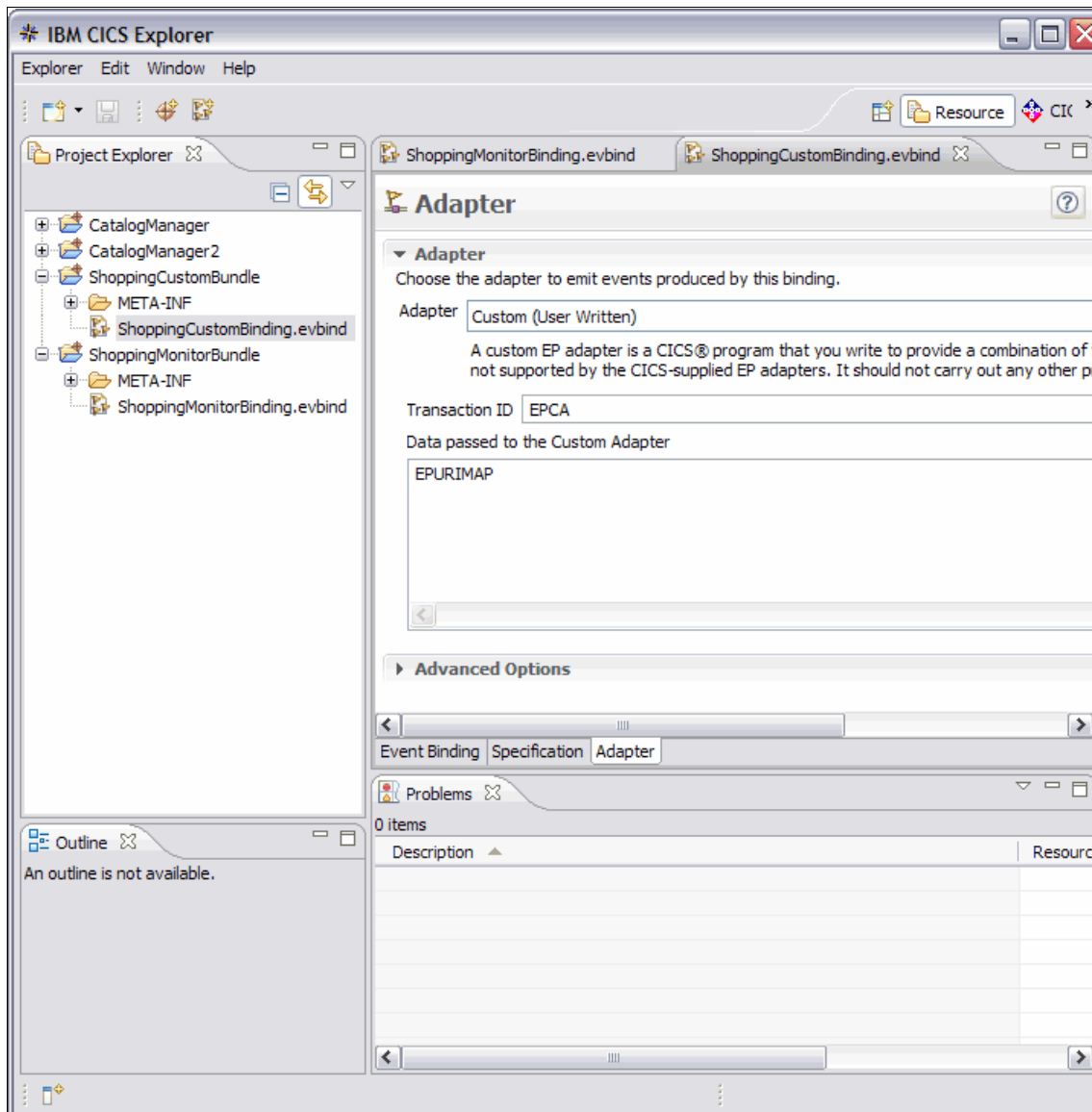


Figure 9-10 Event binding custom EP adapter settings

7. Save ShoppingCustomBinding, right-click **ShoppingCustomBundle**, and choose **Export to System z HFS**. (For more information, see 5.5, “Exporting the bundle project” on page 110)

9.3.3 CICS resource definitions

We use the CICS SM perspective of the CICS Explorer to add the following resource definitions to the resource group EPGRP2 created in 5.6, “Installing the bundle in CICS” on page 114.

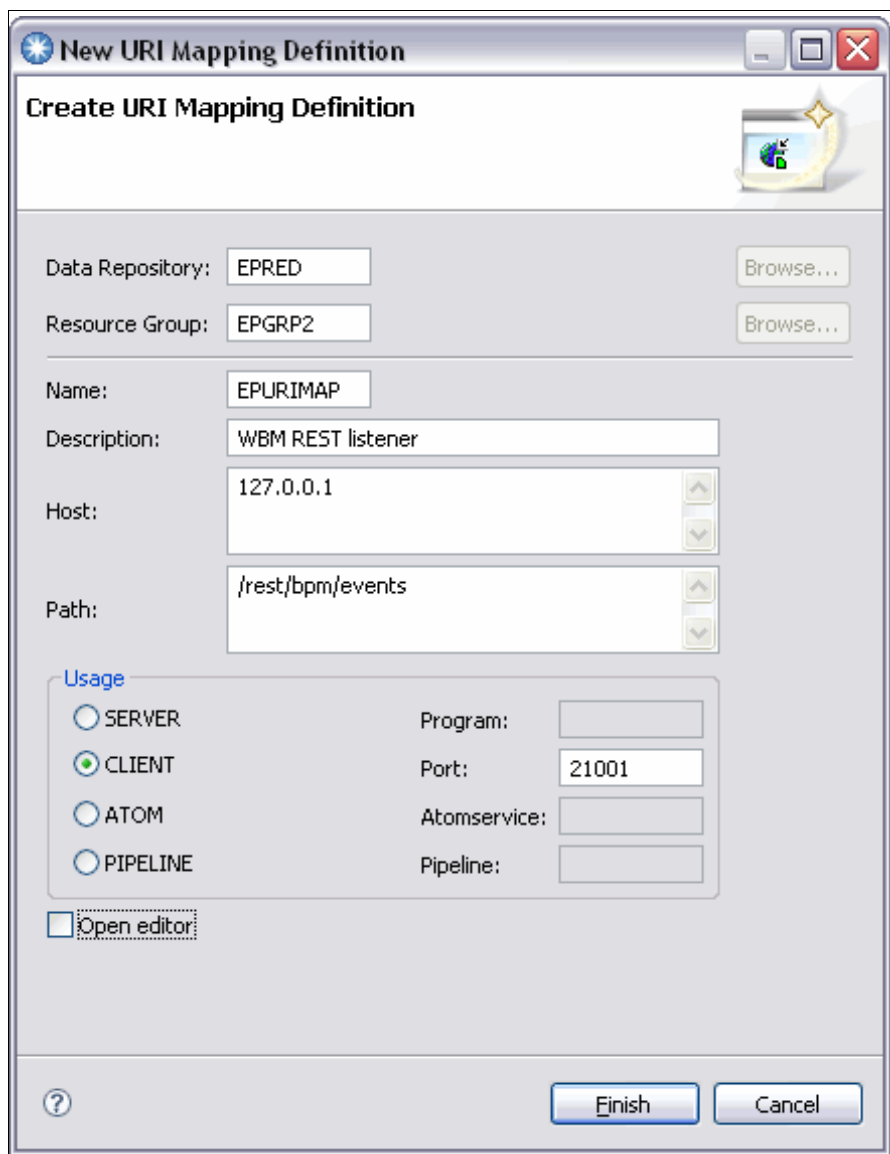
URIMAP

To define and install EPURIMAP as a client URIMAP for the host and port that the WebSphere Business Monitor REST listener is listening on, click

Administration URI Mapping Definitions. See Figure 9-11 on page 328. In the presented view right-click the white space and click **New** and enter the following information:

- ▶ Data Repository: EPRED
- ▶ Resource Group: EPGRP2
- ▶ Name: EPURIMAP
- ▶ Description: WBM REST listener
- ▶ Host: The host name of your WebSphere Business Monitor server
- ▶ Path: /rest/bpm/events

Select the **CLIENT** radio button, supply the port number of your WebSphere Business Monitor REST listener, and click **Finish**.



New URI Mapping Definition

Create URI Mapping Definition

Data Repository: EPRED Browse...

Resource Group: EPGRP2 Browse...

Name: EPURIMAP

Description: WBM REST listener

Host: 127.0.0.1

Path: /rest/bpm/events

Usage

☐ SERVER

☒ CLIENT

☐ ATOM

☐ PIPELINE

Program:

Port: 21001

Atomservice:

Pipeline:

☐ Open editor

? Finish Cancel

Figure 9-11 Create URIMAP resource

You can find the port number by logging on to the administration console of the WebSphere Business Monitor server and checking the port list. It is the WC_defaulthost (or WC_defaulthost_secure).

Right-click the new URIMAP in the URI Mapping Definition view and select **Install**. See Figure 9-12 on page 329.

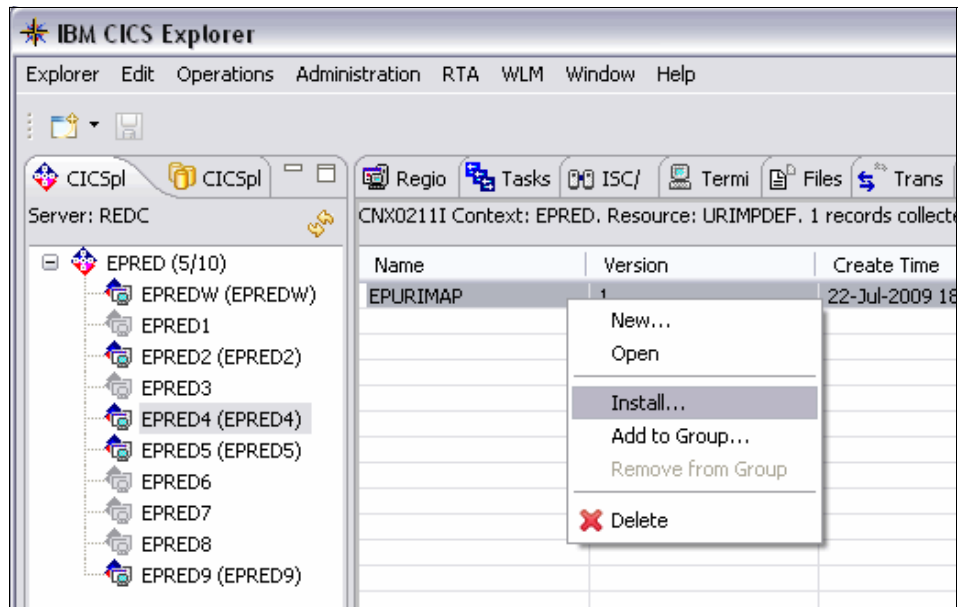


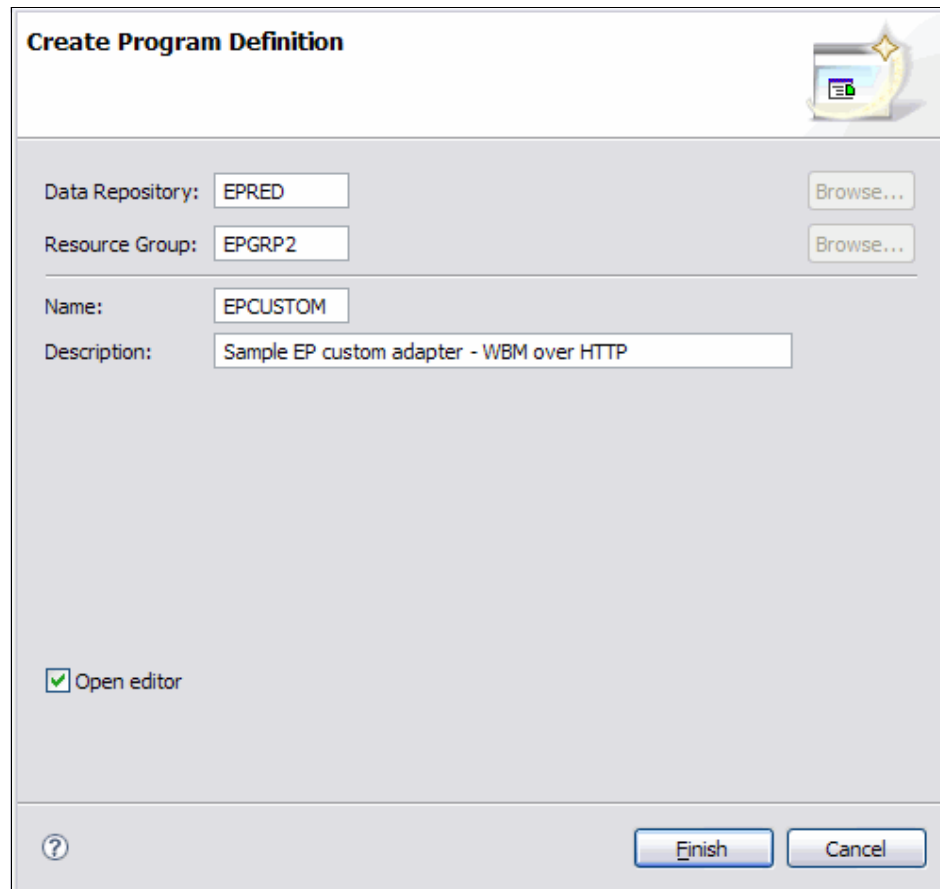
Figure 9-12 Install the URIMAP

PROGRAM

To define and install EPCUSTOM as a CICS PROGRAM, click **Administration Program Definitions**. See Figure 9-13. In the presented view, right-click the white space and click **New** and enter the following information:

- ▶ Data Repository: EPRED
- ▶ Resource Group: EPGRP2
- ▶ Name: EPCUSTOM
- ▶ Description: Sample EP custom adapter - WBM over HTTP

Leave the **Open Editor** check box selected and click **Finish**.



The image shows a 'Create Program Definition' dialog box. It has a title bar with the text 'Create Program Definition' and a small icon of a document with a star. The dialog contains several input fields and buttons. The 'Data Repository' field is set to 'EPRED' with a 'Browse...' button to its right. The 'Resource Group' field is set to 'EPGRP2' with a 'Browse...' button to its right. The 'Name' field is set to 'EPCUSTOM'. The 'Description' field is set to 'Sample EP custom adapter - WBM over HTTP'. At the bottom left, there is a checked checkbox labeled 'Open editor'. At the bottom right, there are two buttons: 'Finish' and 'Cancel'. A help icon (?) is located at the bottom left of the dialog.

Figure 9-13 EPCUSTOM program definition

Then update the following settings in the Program Definition Overview section (Figure 9-14):

- ▶ Language: COBOL
- ▶ Threadsafe: Check the box
- ▶ 31 bit addresses: Check the box

The screenshot shows a web-based interface for defining a program. The window title is "Program Definition (EPCUSTOM) Sample EP custom adapter - WBM ov". The main section is titled "Overview" and contains several tabs: "Basic", "Details", "Storage", "Program reuse", and "User Data". The "Basic" tab is active and shows the following fields: Name: EPCUSTOM, Description: Sample EP custom adap, Version: 1, Created: 13-Jul-2009 08:24:53, Enabled: ☒, and Changed: 13-Jul-2009 08:25:35. The "Details" tab is also visible and shows: Language: COBOL (selected from a dropdown), Non-CICS (Open) API: ☐, Threadsafe (able to use open TCB): ☒, and Display Execution Diagnostic Facility (EDF) screens: ☒. The "Storage" tab shows: Can handle 31 bit addresses (above the 16MB line): ☒, Use Program from the Link Pack Area (LPA): ☐, and Program can write to CICS-key storage: ☐. The "Program reuse" tab shows four radio button options: Reuse if possible (selected), Force reuse, Always load a new copy, and Load a new copy whenever use count drops to zero. The "User Data" tab shows three input fields labeled 1:, 2:, and 3:.

Figure 9-14 Update EPCUSTOM attributes

Right-click the new PROGRAM in the Program Definition view and select **Install**.

Alternatively, you could log on to CICS and use CEDA to copy the definition of the sample custom EP adapter program,

DFHOEPAC:

```
CEDA COPY PROGRAM(DFHOEPAC) GROUP(DFHLEPAG) TO(REDGROUP) AS(EPCUSTOM)
```

TRANSACTION

To define and install EPCA as a CICS TRANSACTION click **Administration Transaction Definitions**. See Figure 9-15 on page 333. In the presented view, right-click the white space and click **New** and enter the following information:

- ▶ Data Repository: EPRED
- ▶ Resource Group: EPGRP2
- ▶ Name: EPCA
- ▶ Description: Sample EP custom adapter
- ▶ Program Name: EPCUSTOM

Leave the **Open Editor** check box selected and click **Finish**.

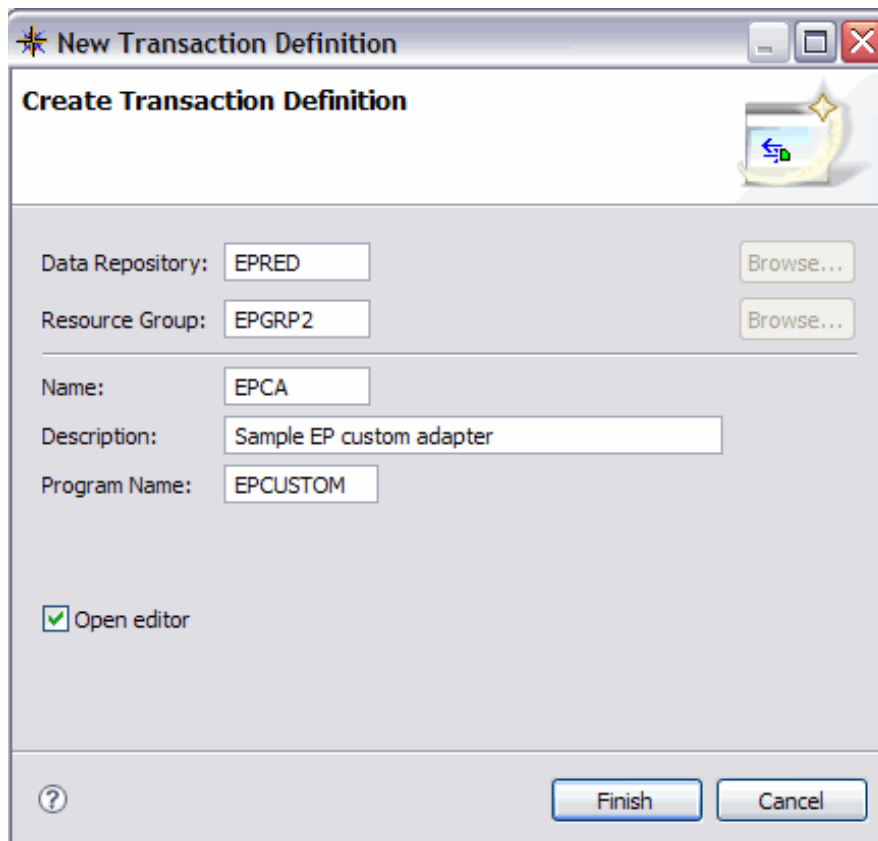


Figure 9-15 EPCA transaction definition

Because there could be a queue of events still to be dispatched at shutdown, we recommend that you define EP adapter transactions as shutdown enabled:

- Select the **Can be run during CICS shutdown** check box in the **Transaction Definition Termination** section. See Figure 9-16.

The screenshot shows a web-based interface for defining transaction attributes. The title bar indicates the context is '*ShoppingEventBinding.' and '*Transaction Definitio'. The main heading is 'Transaction Definition (EPCA) Sample EP custom adapter'. Below this, the 'Termination' section is active, showing three sub-sections: 'Runaway', 'Terminal', and 'In Doubt'. In the 'Terminal' section, the 'Can be run during CICS shutdown' checkbox is checked. The 'In Doubt' section shows 'Specific Time' set to '0:0:0' and 'Backout' selected as the failure action. At the bottom, a navigation bar includes tabs for Overview, Remote, Termination, Runtime, Alias, Recovery, and Attributes.

*ShoppingEventBinding. *Transaction Definitio

Transaction Definition (EPCA) Sample EP custom adapter

Termination

Runaway

Time a task can have processor control before it is assumed to be looping and terminated

☒ Use ICVR System Initialization Parameter

☐ Do not monitor task for runaway condition

☐ Specific Time: (1-2,700,000 ms)

Terminal

If transaction is associated with a terminal, indicate whether the transaction can be run during CICS shutdown

☒ Can be run during CICS shutdown

In Doubt

For region failure, after the unit of work (UOW) has entered the in-doubt period, CICS waits before performing a failure action

☐ Do not wait

☐ Wait for the CICS environment default time

☒ Specific Time: (dd:hh:mm)

The Failure action is applied to recoverable resources if CICS is not waiting to resynchronize with the coordinating region

☒ Backout ☐ Commit

< ||| >

Overview Remote Termination Runtime Alias Recovery Attributes

Figure 9-16 Update EPCA attributes

Right-click the new TRANSACTION in the Transaction Definition view and select **Install**.

Alternatively, you could log on to CICS and use CEDA to copy the definition of TRANSACTION EPAT and change the program name and shutdown enablement setting:

```
CEDA COPY TRANSACTION(EPAT) GROUP(DFH£EPAG) TO(REDGROUP) AS(EPCA)
CEDA ALTER TRANSACTION(EPCA) GROUP(REDGROUP) PROGRAM(EPCUSTOM)
SHUTDOWN(ENABLED)
CEDA INSTALL GROUP(REDGROUP)
```

BUNDLE

To define and install SHOPCUST as a BUNDLE, click **Administration Bundle Definitions**. In the presented view, right-click the white space and click **New** and enter the following information:

- ▶ Data Repository: EPRED
- ▶ Resource Group: EPGRP2
- ▶ Name: SHOPCUST
- ▶ Description: Shopping app with custom adapter

Click **Finish**. See Figure 9-17.

New Bundle Definition

Create Bundle Definition

Data Repository: EPRED Browse...

Resource Group: EPGRP2 Browse...

Name: SHOPCUST

Description: Shopping app with custom adapter

Bundle Directory: /u/cicsrs4/bundles/ShoppingCustomBundle Browse...

☐ Open editor

Finish Cancel

Figure 9-17 Create BUNDLE resource

When we are ready to start generating events from the shopping application, we right-click the new BUNDLE in the Bundle Definition view and select install.

9.3.4 Generating custom events

You are now ready to test that events captured from the sample application are processed correctly by EPCUSTOM:

1. Use the Operations options from the CICS Explorer SM view to check that the resource definitions created above are installed.
2. Log on to CICS and run the MENU sample application as described in 3.2, “Shopping sample application” on page 47.
3. Use transaction CEBR EPURIMAP to check that the expected events have been captured, dispatched to EPCUSTOM and emitted to the expected TS queue.

Figure 9-7 on page 320 gives an example of an event emitted by EPCUSTOM which will be displayed as a single record by the CEBR transaction.

4. Check that the events have been received by WebSphere Business Monitor as described in Chapter 10, “Updating the monitoring to use custom EP adapter events” on page 339.



Updating the monitoring to use custom EP adapter events

In this chapter we discuss monitoring the events that are emitted by the custom EP adapter from Chapter 9, “Custom EP adapters” on page 307.

As with the Chapter 8, “WebSphere Business Monitor scenario” on page 213, we show how to subscribe to, and process, the events in the existing monitor model.

One of the good things about the WebSphere Business Monitor, is that as long as the events that it receives are correctly formatted, it is not aware of, or concerned with, where they originated or what sort of an application created them. This sort of information may be relevant to any monitor model that is consuming them, but to the Monitor it is not relevant.

What that means for us in the context of our scenario is that there is nothing that needs to be additionally configured on the Monitor server itself. We simply need to make our monitor model aware of these events, what they look like, and what to do with them when they are received.

10.1 Create event definitions and event parts

As mentioned in Chapter 9, “Custom EP adapters” on page 307, there are a few basic differences between the events that are emitted by the custom EP adapter and the events that we have been processing so far. This will make a difference as to how we access the payload contained in the custom EP adapter events.

As we know, the application portion of the CBE is in the xs:any slot of the event itself. As we can see in Figure 10-1 and Figure 10-2 on page 341, unlike the CICS events, the custom EP adapter events do not have any of the cics:event information, only the data:payload element.

[-] [e] CommonBaseEvent	
[a] creationTime	2009-07-23T09:24:53.553+00:00
[a] version	1.0.1
[+] [e] sourceComponentId	
[+] [e] situation	
[-] [e] cics:event	
[a] xmlns:cics	http://www.ibm.com/xmlns/prod/cics/events/CBE
[-] [e] cics:context-info	
[e] cics:eventname	QueryStock
[e] cics:usertag	V001
[e] cics:networkapplid	USIBMSC.EPRED4
[e] cics:timestamp	2009-07-16T09:24:53.553+00:00
[e] cics:bindingname	ShoppingMonitorBinding
[e] cics:capturespecname	LINKPOST
[e] cics:UDWid	1910E4E2C9C2D4E2C34BE3C3D7F6F6F0F1F87DF87
[-] [e] cics:payload-data	
[-] [e] data:payload	
[a] xmlns:data	http://www.ibm.com/prod/cics/V001/QueryStock
[e] data:CustomerNumber	+00005
[e] data:StockId	+00006

Figure 10-1 CICS event

[-] [e] CommonBaseEvent	
[a] creationTime	2009-07-23T09:23:27.687Z
[a] extensionName	com.ibm.wbmonitor.EventEmitter
[a] globalInstanceId	CE3F1E0248FDDCBAE9A1DE715372055570
[a] sequenceNumber	192853220953934
[a] version	1.0.1
[+] [e] sourceComponentId	
[+] [e] situation	
[-] [e] data:payload	
[a] xmlns:data	http://www.ibm.com/prod/cics/V001/QueryStock
[e] data:CustomerNumber	000005
[e] data:StockId	000006

Figure 10-2 custom EP adapter event

However, you will notice that both events are using the same schema definitions for the business data payload. This makes adding these new events a straightforward task.

We create new event definitions for each of the events that the custom EP adapter emits. We start with the Query event, which is the one in our diagrams. See Figure 10-3 on page 342.

1. Create a new inbound event (we chose the name CAQuery).
2. Create an event part to point to the payload (we chose eventPayload).
3. Select the query:payload type for this event part and give it a namespace prefix (we chose query).
4. Note that cbe:CommonBaseEvent/query:payload defaults into the path statement. In this instance it is correct, so you can simply accept it.
5. Click **Finish**.

Create New Event Part Type

Create an event part type

Specify the details of the event part. Together, all the event parts describe the structure of the event at run time.

Name:

ID:

Type:

Path:

Figure 10-3 *eventPayload event part*

6. Set a filter condition for this event.

`fn:exists(CAQueryEvent/eventPayload/query:CustomerNumber)` and
`fn:exists(CAQueryEvent/eventPayload/query:StockId)`

Note: We do not have any eventname to access in this event, but we still need to set a filter condition that will evaluate to true for the events that we want.

Each event part describes how to identify and locate itself within the actual event received at run time. An event part has the following key pieces of information:

- ▶ An ID that is used to refer to the event part within path expressions
- ▶ A path that defines how to find, at run time, the content that the event part describes
- ▶ A type that defines the structure of the content identified by the path

This will enable our filter condition to find and check the elements for this event as distinct from any other event. We know that a query event will always have a CustomerNumber and StockId, so we can check for the existence of these elements.

7. Set a correlation expression for this event:

```
Customer_Number =  
fn:substring(CAQueryEvent/eventPayload/query:CustomerNumber,2)
```

Note: As mentioned in 9.2, “How to write a custom EP adapter” on page 308, the custom EP adapter events will not prefix positive numbers with a sign. However, all of the events will prefix the data with - if it has a negative value, so we would be stripping this off for the “not true” numeric elements (such as order number, customer number and so on), that we would not realistically expect to have a negative value.

The event data in the custom EP adapter is replacing that leading + sign with a zero. We need to also strip that off as “00005” and “000005” are not the same thing.

8. Set the event delivery options to be the same as the non-custom event equivalents.
9. Repeat this for each of the other events, creating a CAxxx event for each.
10. Repeat this for the other events in the child contexts.

10.2 Update keys and metrics

As before, any creation event can set the key value for the monitoring context.

1. Set the key values for each of the monitoring contexts to include the new CAXxx events.
Ensure that each of the metrics is populated with the data from these new events.
2. Add a metric mapping expression to each of the metrics in all of the contexts to access the CAXxx event payload.
3. Update counters and stopwatches to include the new events.
4. Update the termination trigger to include the new CASHipOrderEvent.

10.3 Deploy and test monitor model

Rebuild and redeploy the monitor model application.

1. Right-click the server and select **Add and Remove Projects**.
2. Remove the RedbookMMAApplication.

Important: As we are using the development and testing environment, it is not possible simply to deploy a new version of the monitor model application. We must remove it from the server and re-install it. This has the effect of also removing all of the monitoring instance data. In an actual runtime environment we would create a new version of the monitor model to ensure that we retain the previously monitored data.

For more information regarding monitor model versioning, see the InfoCenter.

3. Right-click the monitor model and select **Generate Monitor J2EE Projects**
4. After the application has generated, add it back to the server.
Wait until the monitor model is ready to consume events.
5. Run the sample application to send a stock item query (we chose Customer 000005, as shown in Figure 10-4 on page 345, and Stock Item 000006, as shown in Figure 10-5 on page 345).
6. Log on to the Business Space and navigate to the **Operations Dashboard - Customers**.

We see a stock item query event has been received for customer 000005.

Instances			
<div> <div> <i>i</i> </div> <div>Export ...</div> </div>			
Customer Name	Customer Number	Customer Queries	Stock Queries for Customer
	00005	1	⌵

Figure 10-4 Customer dashboard

Instances	
<div> <div> <i>i</i> </div> <div>Export ...</div> </div>	
Stock Item	
000006	
<div>Drill Up</div>	

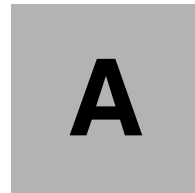
Figure 10-5 Stock item queried

7. Place an order for that customer and item.
8. Ship the order.

We have successfully updated our monitor model and tested the processing with the events from the custom EP adapter. See Figure 10-6.

Instances					
<div> <div> <i>i</i> </div> <div>Export ...</div> <div>⊕ Search for:</div> </div>					
COMPLETED	Order Number	Order Status	Order Quantity	Stock Item	Order Value
★	00027	SHIPPED	10	00006	£69.90
<div>Drill Up</div>					

Figure 10-6 Completed order



Additional material

This book refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG247792>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG247792.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
SG247792.zip	Zipped Code Samples
SG247792doc.zip	Zipped readme

System requirements for downloading the Web material

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 350. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *z/OS: WebSphere Business Process Management V6.2 Production Topologies*, SG24-7733
- ▶ *Business Process Management Enabled by SOA*, REDP-4495

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM Tivoli OMEGAMON XE for CICS Transaction Gateway on z/OS: User's Guide*, SC23-5963

Online resources

These Web sites are also relevant as further information sources:

- ▶ *CICS Transaction Server for z/OS, Version 4 Release 1* infocenter
<http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp>
- ▶ *CICS Explorer* download
<http://www-01.ibm.com/software/http/cics/explorer/>
- ▶ *WebSphere Business Events*
<http://publib.boulder.ibm.com/infocenter/wbevents/v6r2m1/index.jsp>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Implementing Event Processing with CLCS

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Implementing Event Processing with CICS

Instrument your CICS applications for business monitoring

Integrate CICS with WebSphere Business Events

Learn how to extend CICS event processing for your environment

Governance concerns the methods that businesses use to control important aspects of their business and these methods must be measured. Compliance is the tolerance to the criteria that have been set. CICS v4 allows you to configure your system to produce application compliance data feeds and events without application change to show solution compliance on a dashboard.

The governance theme will largely be linked to CICS TS v4.1 which allows you to generate business events without requiring application change. This new capability will allow you to populate business dashboards provided by WebSphere® Business Monitor and to search for patterns in events with WebSphere Business Events. For business event processing, when CICS TS is operating as a stand-alone system, or used in conjunction with WebSphere Business Monitor or WebSphere Business Events (or both), it will enable you to understand and manage your business more easily, and to monitor risk and business compliance more effectively.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks